



VWF Interactive Tools

DEVICE SIMULATION SOFTWARE

SILVACO International
4701 Patrick Henry Drive, Bldg. 1
Santa Clara, CA 95054
Telephone (408) 567-1000
FAX: (408) 496-6080
Internet: www.silvaco.com
E-Mail: support@silvaco.com

July 18, 2005

Notice

The information contained in this document is subject to change without notice.

SILVACO International MAKES NO WARRANTY OF ANY KIND WITH REGARD TO THIS MATERIAL, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTY OF FITNESS FOR A PARTICULAR PURPOSE.

SILVACO INTERNATIONAL shall not be held liable for errors contained herein or for incidental or consequential damages in connection with the furnishing, performance, or use of this material.

This document contains proprietary information, which is protected by copyright laws of the United States. All rights are reserved. No part of this document may be photocopied, reproduced, or translated into another language without the prior written consent of SILVACO INTERNATIONAL.

SIMULATION STANDARD, TCAD DRIVEN CAD, VIRTUAL WAFER FAB, ANALOG ALLIANCE, LEGACY, ATHENA, ATLAS, MERCURY, VICTORY, VYPER, ANALOG EXPRESS, RESILIENCE, DISCOVERY, CELEBRITY, MANUFACTURING TOOLS, AUTOMATION TOOLS, INTERACTIVE TOOLS, TONYPLOT, TONYPLOT3D, DECKBUILD, DEVEDIT, DEVEDIT3D, INTERPRETER, ATHENA INTERPRETER, ATLAS INTERPRETER, CIRCUIT OPTIMIZER, MASKVIEWS, PSTATS, SSUPREM3, SSUPREM4, ELITE, OPTOLITH, FLASH, SILICIDES, MC DEPO/ETCH, MC IMPLANT, S-PISCES, BLAZE/BLAZE3D, DEVICE3D, TFT2D/3D, FERRO, SiGe, SiC, LASER, VCSELS, QUANTUM2D/3D, LUMINOUS2D/3D, GIGA2D/3D, MIXEDMODE2D/3D, FASTBLAZE, FASTLARGESIGNAL, FASTMIXEDMODE, FASTGIGA, FASTNOISE, MOCASIM, SPIRIT, BEACON, FRONTIER, CLARITY, ZENITH, VISION, RADIANT, TWINSIM, UTMOST, UTMOST II, UTMOST III, UTMOST IV, PROMOST, SPAYN, UTMOST IV MEASURE, UTMOST IV FIT, UTMOST IV SPICE MODELING, SMARTSTATS, SDDL, SMARTSPICE, FASTSPICE, TWISTER, BLAST, MIXSIM, SMARTLIB, TESTCHIP, PROMOST-REL, RELSTATS, RELLIB, HARM, RANGER, RANGER3D NOMAD, QUEST, EXACT, CLEVER, STELLAR, HIPEX-NET, HIPEX-R, HIPEX-C, HIPEX-RC, HIPEX-CRC, EM, POWER, IR, SI, TIMING, SN, CLOCK, SCHOLAR, EXPERT, SAVAGE, SCOUT, DRAGON, MAVERICK, GUARDIAN, ENVOY, LISA, EXPERTVIEWS, and SFLM are trademarks of SILVACO INTERNATIONAL.

All other trademarks mentioned in this manual are the property of their respective owners.

© 1990, 1991, 1992, 1993, 1994, 1995, 1996, 1997, 1998, 2000, 2002, 2004, 2005 by **SILVACO** International Inc.

Reader Comment Sheet

We welcome your evaluation of this manual. Your comments and suggestions help us to improve our publications. If you have any responses to the questions below, please let us know. Please write your observations down and send your complaints, bug reports, suggestions or comments to the e-mail address listed below.

- Is this manual technically accurate?
- Are the concepts and wording easy to understand?
- Is the size of this manual convenient for you?
- Is the manual's arrangement convenient for you?
- Do you consider this manual to be easily readable?

Please add any additional relevant comments and fax your comments to:

SILVACO International

Attention: Technical Publications

1701 Patrick Henry Drive, Building 1

Santa Clara, CA 95054

You can also e-mail us at support@silvaco.com or visit our website at <http://www.silvaco.com>.

How to Read this Manual

Style Conventions		
Font Style/Convention	Description	Example
•	This represents a list of items or terms.	<ul style="list-style-type: none"> Bullet A Bullet B Bullet C
1. 2. 3.	This represents a set of directions to perform an action.	To open a door: <ol style="list-style-type: none"> 1. Unlock the door by inserting the key into keyhole. 2. Turn key counter-clockwise. 3. Pull out the key from the keyhole. 4. Grab the doorknob and turn clockwise and pull.
→	This represents a sequence of menu options and GUI buttons to perform an action.	File→Open
Courier	This represents the commands, parameters, and variables syntax.	HAPPY BIRTHDAY
New Century Schoolbook Bold	This represents the menu options and buttons in the GUI.	File
<i>New Century Schoolbook Italics</i>	This represents the equations.	$abc=xyz$
Note:	This represents the additional important information.	Note: Make sure you save often while running an experiment.
NEW CENTURY SCHOOLBOOK IN SMALL CAPS	This represents the names of the SILVACO/SIMUCAD AUTOMATION DESIGN Products.	ATHENA, ATLAS, EXPERT, GATEWAY, HIPEX, SMART-SPICE, STELLAR, and UTMOST.

Table of Contents

Chapter 1

Introduction	1-1
1.1: Getting Started	1-1
1.2: Interactive Tools	1-3
1.2.1: DeckBuild	1-3
1.2.2: TonyPlot	1-4
1.2.3: DevEdit	1-4
1.2.4: MaskViews	1-4
1.2.5: Optimizer	1-4
1.2.6: Manager	1-5
1.2.7: SPDB	1-5
1.3: VWF Automation Tools	1-6
1.3.1: The Virtual Wafer FAB	1-6
1.3.2: VWF Experimentation	1-6
1.3.3: Advanced Features	1-7
1.3.4: Productivity Enhancements	1-8

Chapter 2

Tutorial	2-1
2.1: Overview	2-1
2.2: Introduction	2-2
2.3: Getting Started	2-3
2.4: Selecting and Loading An Example	2-5
2.5: Process Simulation	2-7
2.5.1: Running a Simulation	2-7
2.5.2: Plotting A Structure	2-8
2.5.3: Using TonyPlot	2-9
2.5.4: Using The History Function	2-12
2.5.5: Comparing Structure Files	2-13
2.5.6: Overlaying Two Plots	2-15
2.5.7: Continuing the Process Simulation	2-15
2.5.8: Plotting 2-D Structures	2-16
2.5.9: Contour Plots	2-17
2.5.10: Interactive Outline Plots	2-18
2.5.11: Extracting Process Parameters	2-19
2.6: Completing The Device Structure	2-24
2.7: Device Simulation	2-25
2.7.1: Defining The Device Physics	2-25
2.7.2: Performing The Device Simulation	2-25
2.7.3: Extracting Device Parameters	2-27
2.8: Going Further	2-28

Chapter 3

Manager	3-1
3.1: Starting Manager	3-1
3.2: Using Manager	3-2
3.2.1: File and Application Windows	3-2
3.2.2: Changing the Current Directory	3-2
3.2.3: Starting Applications	3-2
3.2.4: On Line Help	3-2
3.2.5: Customizing	3-3
3.2.6: Applications	3-4
3.2.7: File types	3-5

Chapter 4

DeckBuild	4-1
4.1: Introduction	4-1
4.1.1: Overview	4-1
4.1.2: DeckBuild's Purpose	4-1
4.1.3: Features	4-1
4.2: QuickStart	4-5
4.2.1: Overview	4-5
4.2.2: Starting DeckBuild	4-5
4.2.3: Writing a SSUPREM3 Input Deck	4-5
4.2.4: Running A Deck	4-8
4.2.5: Quitting DeckBuild	4-10
4.3: Invoking DeckBuild	4-11
4.3.1: Syntax	4-11
4.3.2: Description	4-11
4.3.3: Options	4-11
4.4: DeckBuild Controls	4-14
4.4.1: Main Window Layout	4-14
4.4.2: Text & TTY Subwindows	4-15
4.4.3: Using the Text Subwindow	4-15
4.4.4: Using the TTY Subwindow	4-18
4.5: Main Control	4-20
4.5.1: Control Pad	4-20
4.5.2: Choosing a Simulator	4-20
4.5.3: Simulator Properties	4-21
4.5.4: Start Simulator	4-21
4.5.5: Simulator Controls	4-21
4.5.6: Options Category	4-23
4.5.7: Messages Category	4-25
4.5.8: Formatting Category	4-26
4.5.9: Arguments Category	4-26
4.6: Execution Control	4-27
4.6.1: Execution Concepts	4-27
4.6.2: Execution Control Buttons	4-28
4.6.3: Stepping Through and Running the Deck	4-28
4.6.4: Setting and Clearing Breakpoints	4-28
4.6.5: Setting the Current Line	4-28
4.6.6: Pausing, Stopping, and Restarting the Simulator	4-29
4.6.7: Initializing the Simulator	4-29
4.7: Commands	4-30

4.7.1: Deck Writing Paradigm	4-30
4.7.2: Parsing the Deck	4-30
4.7.3: Process Simulators	4-31
4.7.4: Mercury Tool	4-32
4.7.5: Clever	4-32
4.8: Tools	4-33
4.8.1: Starting TonyPlot	4-33
4.8.2: Starting Maskviews	4-34
4.8.3: Starting Text Editor	4-37
4.8.4: Starting Manager	4-37
4.9: History	4-38
4.9.1: Overview	4-38
4.9.2: History Control	4-38
4.10: Auto Interfacing	4-40
4.10.1: Overview	4-40
4.10.2: Scenario	4-40
4.11: IC Layout Interface	4-43
4.11.1: Overview	4-43
4.11.2: Creating a Generic Deck	4-43
4.11.3: Regions	4-44
4.11.4: Rules of Thumb	4-46
4.11.5: Mask Bias, Misalignment, and Delta CD	4-46
4.11.6: Using DevEdit with IC Layout	4-47
4.12: UTMOST Interface	4-48
4.12.1: Overview	4-48
4.12.2: Setting Up An UTMOST Input Deck	4-48
4.13: SmartSpice Interface	4-56
4.14: Internal Interface	4-57
4.15: Remote Simulation	4-58
4.15.1: Overview	4-58
4.15.2: Remote Options	4-58
4.15.3: Troubleshooting	4-58
4.16: Statements	4-60
4.16.1: Overview	4-60
4.16.2: DeckBuild Commands	4-60
4.16.3: ASSIGN	4-60
4.16.4: AUTOELECTRODE	4-63
4.16.5: DEFINE and UNDEFINE	4-64
4.16.6: EXTRACT	4-65
4.16.7: GO	4-65
4.16.8: IF, ELSE and IF.END	4-67
4.16.9: LOOP, L.END and L.MODIFY	4-67
4.16.10: MASK	4-68
4.16.11: MASKVIEWS	4-69
4.16.12: SET	4-70
4.16.13: SOURCE	4-72
4.16.14: STMT	4-73
4.16.15: SYSTEM	4-74
4.16.16: TONYPLOT	4-75
4.17: Environment Variables	4-76
4.18: Error Messages	4-77
4.18.1: Text Subwindow Error Messages	4-77
4.18.2: TTY Subwindow Error Messages	4-77

Chapter 5

DeckBuild:Extract	5-1
5.1: Overview	5-1
5.2: Process Extraction	5-2
5.2.1: Entering a Process Extraction Statement	5-4
5.2.2: Extracting a Curve	5-5
5.3: Customized Extract Statements	5-7
5.3.1: Extract Syntax	5-7
5.3.2: DEFAULTS	5-20
5.3.3: Examples of Process Extraction	5-21
5.4: Device Extraction	5-29
5.4.1: The Curve	5-29
5.4.2: Curve Manipulation	5-31
5.4.3: BJT Example	5-32
5.5: General Curve Examples	5-33
5.5.1: Curve Creation	5-33
5.5.2: Min Operator with Curves	5-33
5.5.3: Max Operator with Curves	5-33
5.5.4: Ave Operator with Curves	5-33
5.5.5: X Value Intercept for Specified Y	5-33
5.5.6: Y Value Intercept for Specified X	5-33
5.5.7: Abs Operator with Axis	5-34
5.5.8: Min Operator with Axis Intercept	5-34
5.5.9: Max Operator with Axis Intercept	5-34
5.5.10: Second Intercept Occurrence	5-34
5.5.11: Gradient at Axis Intercept	5-34
5.5.12: Axis Manipulation with Constants	5-34
5.5.13: X Axis Interception of Line Created by Maxslope Operator	5-34
5.5.14: Y Axis Interception of Line Created by Minslope Operator	5-35
5.5.15: Axis Manipulation Combined with Max and Abs Operators	5-35
5.5.16: Axis Manipulation Combined with Y Value Intercept	5-35
5.5.17: Derivative	5-35
5.5.18: Data Format File Extract with X Limits	5-35
5.5.19: Impurity Transform against Depth	5-35
5.6: MOS Device Tests	5-36
5.7: Extracted Results	5-37
5.7.1: Units	5-37
5.8: Extract Features	5-38
5.8.1: Extract Name	5-38
5.8.2: Variable Substitution	5-38
5.8.3: Min and Max Cutoff Values	5-38
5.8.4: Multi-Line Extract Statements	5-39
5.8.5: Extraction and the Database (VWF)	5-39
5.9: QUICKBIP Bipolar Extract	5-40
5.10: Using Extract with ATLAS	5-43

Chapter 6

DeckBuild: Optimizer	6-1
6.1: Overview	6-1
6.1.1: Features	6-1
6.1.2: Terminology	6-1
6.2: Using The Optimizer	6-2
6.2.1: Overview	6-2
6.2.2: The Optimizer Window	6-2
6.2.3: Optimization Modes	6-3
6.2.4: Optimizing	6-3
6.3: Parameters	6-4
6.3.1: Adding Parameters	6-4
6.3.2: Deleting Parameters	6-6
6.3.3: Editing A Parameter	6-7
6.3.4: Linked Parameters	6-9
6.3.5: Parameter Defaults	6-10
6.3.6: Copying Parameters To The Deck	6-10
6.3.7: Enabling/Disabling Parameters	6-10
6.3.8: Folding Columns	6-11
6.4: Targets	6-12
6.4.1: Adding A Target	6-12
6.4.2: Deleting A Target	6-16
6.4.3: Editing A Target	6-17
6.4.4: Enabling/Disabling Target	6-18
6.4.5: Folding Columns	6-19
6.5: Setup	6-20
6.5.1: Overview	6-20
6.5.2: Editing Setup Values	6-20
6.5.3: Saving The Setup	6-20
6.6: Graphics	6-21
6.6.1: Overview	6-21
6.7: Results	6-23
6.7.1: Overview	6-23
6.7.2: Sensitivity	6-23
6.8: Worksheet Editing	6-24
6.8.1: Overview	6-24
6.8.2: Numeric Values	6-24
6.8.3: Selecting Rows	6-24
6.8.4: Mouseless Operation	6-25
6.9: File I/O	6-26
6.9.1: Overview	6-26
6.9.2: Creating A New File	6-26
6.9.3: Working With An Existing File	6-27
6.9.4: Saving The Existing File	6-27
6.10: Printing	6-28
6.10.1: Overview	6-28
6.11: Optimization Tuning	6-30

Chapter 7

TonyPlot	7-1
7.1: Overview	7-1
7.1.1: Examining Data	7-1
7.1.2: Online Help	7-2
7.1.3: Terminology	7-2
7.1.4: Standard Controls	7-2
7.2: Invoking	7-4
7.2.1: Overview	7-4
7.2.2: Starting	7-4
7.3: The Base Window	7-7
7.3.1: Overview	7-7
7.3.2: File Menu	7-7
7.3.3: View Menu	7-9
7.3.4: Plot Menu	7-9
7.3.5: Tools Menu	7-9
7.3.6: Print Menu	7-10
7.3.7: The Production Menu	7-10
7.3.8: Properties Menu	7-11
7.3.9: Help Menu	7-11
7.4: The File Loader	7-12
7.4.1: Loading files	7-12
7.4.2: Changing Directories	7-12
7.4.3: File Filtering	7-12
7.4.4: File Options	7-12
7.5: Plot Control	7-13
7.5.1: Plot Selection	7-13
7.5.2: Pointer Zooming	7-13
7.6: Key Commands	7-14
7.7: Command Stream	7-15
7.8: The Plot Menu	7-16
7.9: 3D Plot Control	7-18
7.9.1: Rotation	7-18
7.9.2: Scaling	7-18
7.10: Plot Display	7-19
7.10.1: 2D Mesh Plot Display	7-19
7.11: X-Y Graph Display	7-26
7.11.1: Cartesian Graphs	7-26
7.11.2: Polar Charts	7-27
7.11.3: Smith Charts	7-27
7.11.4: Cross Section Display	7-28
7.12: RSM Display	7-29
7.12.1: 1D RSM Graphs	7-29
7.12.2: 2D RSM Contours	7-30
7.12.3: 3D RSM Surfaces	7-30
7.13: Statistics Display	7-31
7.13.1: Histograms	7-31
7.13.2: Pie Charts	7-32
7.13.3: Scatter Plot	7-32
7.13.4: Box Plot	7-32
7.13.5: Sunray Plot	7-32
7.14: Annotation	7-33

7.14.1: Titles	7-33
7.14.2: Show	7-34
7.14.3: Range	7-34
7.14.4: Statistics Plots	7-34
7.14.5: Axis Labels	7-34
7.14.6: Footers	7-34
7.14.7: Special Characters and Macros	7-34
7.15: Labels	7-35
7.15.1: Control Items	7-35
7.15.2: Placing Labels	7-36
7.15.3: Special Labels	7-37
7.16: Tools	7-38
7.16.1: Outline	7-38
7.16.2: Ruler	7-40
7.16.3: Probe	7-41
7.16.4: Movie	7-42
7.16.5: HP 4145 Emulator	7-43
7.16.6: Integrate	7-44
7.16.7: Tracers	7-46
7.16.8: Poisson Solver	7-47
7.17: Printing	7-49
7.17.1: Print Options	7-49
7.17.2: Printer Editor	7-50
7.17.3: Form Editor	7-51
7.17.4: Printing At Startup	7-52
7.17.5: Queues and Printers	7-52
7.17.6: System Configuration	7-52
7.17.7: Adding Printers To TonyPlot	7-52
7.17.8: Adding Forms To TonyPlot	7-53
7.17.9: Setting Print Options	7-54
7.18: Printer Drivers	7-55
7.18.1: Stack Size	7-55
7.19: Properties	7-56
7.19.1: Drawing Options	7-56
7.19.2: Plot Options	7-57
7.19.3: Main Window	7-58
7.19.4: Tool Settings	7-59
7.19.5: Overlays	7-60
7.19.6: General Colors	7-61
7.19.7: Structure Colors	7-62
7.19.8: Sequence Colors	7-63
7.19.9: Sequence Marks	7-64
7.19.10: Key Options	7-64
7.19.11: Environment	7-65
7.19.12: Fonts	7-66
7.19.13: Miscellaneous	7-67
7.19.14: Materials	7-68
7.19.15: Functions	7-69
7.20: Key Legends	7-70
7.20.1: Overview	7-70
7.20.2: Types Of Keys	7-70
7.20.3: Positioning Key Boxes	7-70
7.20.4: Drawing Styles	7-71
7.21: The Command Stream	7-72

7.21.1: Help	7-72
7.21.2: Finishing TPCS	7-72
7.22: Functions	7-76
7.22.1: Use Of Functions	7-76
7.22.2: Defining Functions	7-76
7.22.3: Plotting	7-76
7.22.4: Function Macros	7-77
7.22.5: Function Syntax	7-78
7.22.6: Function In TPCS	7-79
7.23: User Data Files	7-80
7.23.1: Overview	7-80
7.23.2: Loading User Data Files	7-80
7.23.3: Creating User Data Files	7-80
7.23.4: Data Format	7-80
7.23.5: Examples	7-81
7.24: Set Files	7-82
7.24.1: Creating	7-82
7.24.2: Loading	7-82
7.24.3: Setfile Syntax	7-83
7.25: Overlays	7-84
7.25.1: Making An Overlay	7-84
7.25.2: Splitting An Overlay	7-84
7.25.3: Overlay Control	7-84
7.25.4: Overlay Display	7-84
7.25.5: Identifying Data	7-84
7.25.6: Level Names	7-85
7.25.7: Cutlines	7-85
7.25.8: Properties	7-85
7.26: Production Mode	7-86
7.26.1: Outline	7-86
7.26.2: Enabling Production Mode	7-86
7.26.3: The Production Mode Popup	7-86
7.26.4: Interactive RSM Control	7-87
7.26.5: The Input Sliders	7-87
7.26.6: Failure Analysis	7-87
7.26.7: Synthesis	7-88
7.26.8: Yield Analysis	7-89
7.26.9: Input Parameter Ranges	7-89
7.26.10: Input Distributions	7-90
7.26.11: SPC Limits	7-91
7.26.12: Experimental Results	7-91
7.26.13: Optimizer Setup	7-92
7.26.14: ASA Setup	7-93

Chapter 8

TonyPlot3D	8-1
8.1: Overview	8-1
8.2: Differences Between TonyPlot3D and TonyPlot	8-2
8.2.1: Starting TonyPlot3D	8-2
8.3: Main Window	8-3
8.3.1: Main Menu Options	8-4
8.3.2: Main Toolbar Options	8-6
8.3.3: Plot Control Toolbar Options	8-7

8.3.4: Plot Control Using the Mouse	8-7
8.4: Display Modes	8-9
8.4.1: Regions	8-11
8.4.2: Contours	8-12
8.4.3: Rays	8-14
8.4.4: Isosurface	8-16
8.4.5: Vectors	8-18
8.5: Tools	8-20
8.5.1: Object Editor	8-20
8.5.2: Cutplane	8-26
8.5.3: Probe	8-30
8.5.4: Ruler	8-32
8.6: Properties	8-34
8.6.1: Camera	8-34
8.6.2: Color	8-35
8.6.3: Fonts	8-35
8.6.4: Legend	8-36
8.6.5: Lights	8-37
8.6.6: Mouse	8-38
8.6.7: Structure	8-39
8.7: Operating Platforms	8-40
8.7.1: SunOS 5.x UltraSPARC and SPARCstations	8-40
8.7.2: HP 9000/7xx Workstations	8-40
8.7.3: Linux RedHat (PC Compatibles)	8-41
8.7.4: Windows NT/2000/XP (PC Compatibles)	8-42
Chapter 9	
DevEdit	9-1
9.1: Overview	9-1
9.1.1: The Problem	9-1
9.1.2: The Solution	9-1
9.1.3: When to Use DevEdit	9-1
9.1.4: When Not to Use DevEdit	9-1
9.1.5: Getting Started	9-2
9.2: Base Window	9-3
9.2.1: Layout and Functionality	9-3
9.2.2: Control Panel	9-4
9.2.3: Main Panel Controls	9-4
9.2.4: Control Windows	9-4
9.3: FILE CONTROL	9-5
9.3.1: Using Devedit	9-5
9.3.2: Loading a Silvaco Standard Structure File	9-5
9.3.3: Saving a Silvaco Standard Structure File	9-5
9.3.4: Difference - Silvaco Standard vs. Devedit	9-5
9.3.5: Loading a Command File	9-6
9.3.6: Default Files	9-6
9.4: Tutorial	9-7
9.4.1: Goal And Purpose Of Creating A New Mesh	9-7
9.4.2: EXAMPLE 1 - CREATE A NEW STRUCTURE	9-8
9.4.3: EXAMPLE 2 - REMESHING AN EXISTING STRUCTURE	9-20
9.4.4: Advanced Features	9-29
9.5: STRUCTURE EDITING	9-34
9.5.1: Overview	9-34

9.5.2: Selecting The Resolution	9-34
9.6: EDITING REGIONS	9-35
9.6.1: Adding a Region	9-35
9.6.2: Selecting Region Material	9-35
9.7: DRAWING REGIONS	9-36
9.7.1: Overview	9-36
9.7.2: Setting Base Impurity	9-37
9.7.3: Deleting Regions	9-37
9.7.4: Modify Regions	9-38
9.7.5: Deleting Boundary Points	9-38
9.8: DOPING DEFINITION	9-39
9.8.1: Overview	9-39
9.8.2: Defining an Impurity Source Line	9-39
9.8.3: Defining an Impurity Source Box	9-39
9.8.4: Doping Source Attributes	9-40
9.8.5: Deleting Source Objects	9-40
9.8.6: 3D Doping	9-40
9.9: MESHBUILD MESHING	9-41
9.9.1: Overview	9-41
9.9.2: Boundary Conditioning	9-41
9.9.3: Limitations	9-42
9.9.4: Mesh Constraints	9-42
9.9.5: Adaptive Meshing	9-43
9.9.6: Refinement	9-43
9.9.7: Manually Refining The Mesh	9-44
9.9.8: Manually Relaxing A Mesh	9-44
9.9.9: Tensor Product Mesh	9-44
9.9.10: Work Area Resizing	9-44
9.9.11: Defining 3D Structures	9-44
9.10: CREATING A NEW MESH	9-45
9.10.1: Setting The Mesh Controls	9-45
9.10.2: Base Mesh Parameters	9-45
9.10.3: Refining On Impurities	9-45
9.10.4: Mesh Constraints	9-45
9.10.5: Final Meshing	9-45
9.10.6: Saving the Silvaco Standard Structure File	9-45
9.11: IMPURITIES	9-46
9.11.1: Viewing Impurities	9-46
9.11.2: Impurity Definition	9-46
9.11.3: Impurities Loaded From A Structure	9-46
9.11.4: Add Impurity Mode	9-46
9.11.5: Defining An Impurity Source Area	9-47
9.11.6: Defining Impurity Roll-off Direction	9-47
9.12: ROLL-OFF FUNCTION	9-51
9.12.1: Analytic Functions	9-51
9.12.2: Doping Profiles	9-53
9.12.3: Join Function	9-54
9.12.4: Deleting Impurities	9-55
9.12.5: Editing Impurities	9-55
9.12.6: Combining Impurity Rolloffs	9-55
9.13: STATEMENTS	9-59
9.13.1: Overview	9-59
9.13.2: Cards And Parameters	9-59
9.13.3: BASE.MESH	9-60

9.13.4: BOUNDARY.CONDITIONING	9-60
9.13.5: CONSTRAINT.MESH	9-62
9.13.6: CUT	9-65
9.13.7: DEPOSIT	9-65
9.13.8: FLIP	9-66
9.13.9: IMPURITY	9-67
9.13.10: IMPURITY REFINER	9-71
9.13.11: INITIALIZE	9-72
9.13.12: JOIN	9-73
9.13.13: MESH	9-73
9.13.14: MIRROR	9-74
9.13.15: MOVE	9-75
9.13.16: PROFILE	9-76
9.13.17: QUIT	9-77
9.13.18: REFINER	9-77
9.13.19: REGION	9-78
9.13.20: RENUMBER.REGIONS	9-80
9.13.21: SOURCE	9-80
9.13.22: STRETCH	9-80
9.13.23: STRUCTURE	9-83
9.13.24: SUBSTRATE	9-84
9.13.25: WORK.AREA	9-84
9.13.26: Z.PLANES	9-85
9.13.27: GENERIC PARAMETER - BOOLEAN TYPE	9-86
9.13.28: GENERIC PARAMETER - COLOR	9-86
9.13.29: GENERIC PARAMETER - IMPURITY	9-87
9.13.30: GENERIC PARAMETER - MATERIAL	9-94
9.13.31: GENERIC PARAMETER - PATTERN	9-99

Chapter 10

MaskViews	10-1
10.1: Introduction	10-1
10.2: Starting	10-2
10.2.1: From DeckBuild	10-2
10.2.2: Using MaskViews Inside The VWF	10-4
10.2.3: Starting MaskViews From The UNIX Prompt	10-4
10.3: MaskViews Main Window	10-5
10.3.1: Layout and Functionality	10-5
10.4: Editing	10-7
10.4.1: Defining Edit Parameters	10-7
10.4.2: Defining Layout Layers	10-8
10.4.3: Drawing Objects	10-9
10.4.4: User Defined Objects	10-10
10.4.5: Polygon Editing	10-11
10.4.6: Object Editing	10-12
10.5: Simulator Control	10-13
10.5.1: Overview	10-13
10.5.2: SSUPREM3	10-13
10.5.3: ATHENA	10-13
10.5.4: OPTOLITH	10-17
10.6: Files	10-18
10.6.1: Loading and Saving Files	10-18
10.6.2: Viewing Outline Files	10-19
10.6.3: GDSII & CIF Import/Export	10-19

10.6.4: Creating GDSII Stream Format	10-21
10.6.5: Viewing OPTOLITH Image File	10-21
10.7: Utilities	10-22
10.7.1: Regions	10-22
10.7.2: Rescaling	10-23
10.7.3: Zoom and Pan	10-23
10.7.4: Reordering Layers	10-24
10.7.5: Printouts	10-25
10.7.6: On-Line Help	10-25
10.7.7: Release Documentation	10-25
10.8: Properties	10-26
10.8.1: Overview	10-26
10.8.2: Default Properties	10-26
10.8.3: Display Properties	10-27
10.8.4: 3D Mask Properties	10-27
10.8.5: Drag and Drop	10-27
10.8.6: Printer Properties	10-28
10.8.7: Import Properties	10-28
10.8.8: Color Properties	10-28
10.8.9: Notes On Monochrome Operation	10-28

Appendix A

Models and Algorithms A-1

A.1: Introduction	A-1
A.1.1: Physical Models	A-1
A.2: Concentration Dependent Mobility	A-2
A.3: Field Dependent Mobility Model	A-3
A.4: Sheet Resistance Calculation	A-4
A.5: Threshold Voltage Calculation	A-5
A.5.1: Breakdown Voltage Calculation	A-5

Appendix B

DBInternal B-1

B.1: DBInternal	B-1
B.1.1: Example	B-1
B.2: The Template File	B-3
B.3: The Experiment File	B-4
B.3.1: Load command	B-4
B.3.2: Experiment command	B-4
B.3.3: Save Command	B-4
B.4: Technical Details	B-5
B.5: DBInternal Commands	B-6
B.5.1: doe	B-6
B.5.2: endsave	B-8
B.5.3: log	B-8
B.5.4: monte_carlo	B-8
B.5.5: no_exec	B-10
B.5.6: option	B-11
B.5.7: save	B-11
B.5.8: sweep	B-12

1.1: Getting Started

When starting for the first time, you are directed to use the DECKBUILD tool. From here, the use of software will unfold. See Chapter 4: “DeckBuild” and the ATHENA USER’S MANUAL for a tutorial guide on how to use DECKBUILD.

The VIRTUAL WAFER FAB (VWF) FRAMEWORK (Figure 1-1) is comprised of three basic component sets:

1. VWF CORE TOOLS: These tools simulate either a semiconductor device being processed or a semiconductor device being tested electrically. The VWF CORE TOOLS are ATHENA, ATLAS, and SSUPREM3.
2. VWF INTERACTIVE TOOLS: These tools are designed to be used interactively in the construction of a single input deck. Being Graphical User Interface (GUI) based, they make the job of constructing an input deck more efficient. The interactive tools can be used either in conjunction with a set of files or as integral components of the surrounding VWF AUTOMATION TOOLS.
3. VWF AUTOMATION TOOLS: These tools enable you to perform large scale experimental studies to create results for subsequent statistical analysis. The VWF AUTOMATION TOOLS make use of both distributed database technology and interprocess communications (IPC) software methods.

This manual describes the use of the VWF INTERACTIVE TOOLS. The implied methodology used is that a simulation can be created interactively and tuned to a known result. Once complete, you may seek to understand the performance of a device in cases where the manufacturing process varies. The VWF AUTOMATION TOOLS can then be used to conduct large simulation based experimental split lots. Section 1.3: “VWF Automation Tools” briefly describes the VWF AUTOMATION TOOLS.

The basic concept of VWF incorporates the use of large parallel machines, or networks of workstations. VWF then allows you to use AUTOMATION TOOLS to create simulation results in sufficient quantity for use in design optimization.

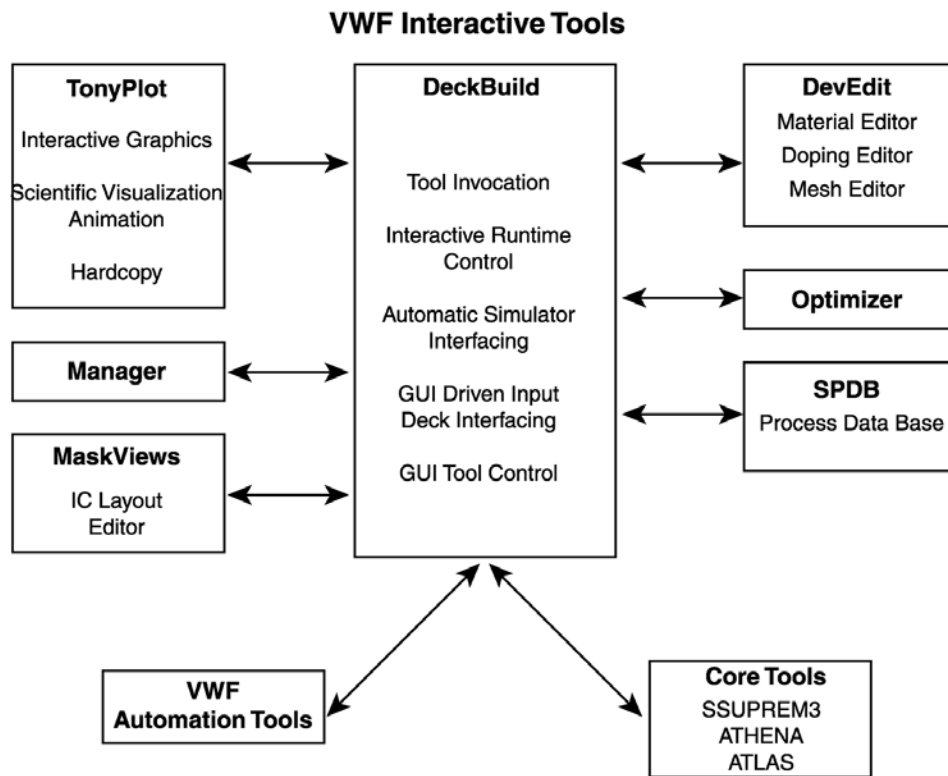


Figure 1-1: Virtual Wafer Fab (VSF) Framework

1.2: Interactive Tools

The VWF INTERACTIVE TOOLS provides a versatile environment for using Semiconductor Technology CAD (TCAD) tools. It does this in two ways. First, making it easier to use process and device simulators. Second, automating simulation tasks associated with developing new technologies.

Tasks that must be performed by all physically based process and device simulators are readily identified. For example, structure specification, meshing, and data visualization, are common to all simulators. Major benefits are obtained by implementing these common functions as specialized tools that are accessed as required by individual simulators. You benefit from the resulting consistency in the way tasks are performed, which makes it much easier to learn and use simulators. In addition, new simulators can be developed more rapidly and are smaller. The VWF INTERACTIVE TOOLS of the SILVACO Framework have been implemented using modern software engineering techniques.

The INTERACTIVE TOOLS communicate with each other and with process and device simulators through a Standard Structure Format (SSF). The block diagram (Figure 1-1) indicates the INTERACTIVE TOOLS and their relationships to the process and device simulators.

VWF INTERACTIVE TOOLS users acquire all, or a subset, of the SILVACO Framework conforming simulators. These simulators include SSUPREM3, ATHENA, and ATLAS. ATHENA is Silvaco's 2D process simulation system. It can be used with any combination of OPTOLITH, ELITE, and SSUPREM4. ATLAS is Silvaco's 2D device simulation system. It is used by SPISCES or BLAZE, and any combination of the other ATLAS products (GIGA, LUMINOUS, TFT, and MIXEDMODE). Individual process and device simulators reside in a Simulator Library and are invoked as required by DECKBUILD. The following information provides a brief description of each VWF INTERACTIVE TOOL.

1.2.1: DeckBuild

The Interactive Runtime Environment is the central environment for interactively using process and device simulators. It provides many important capabilities. A GUI for input deck specification allows you to avoid simulator-specific input syntax. Information is typed into a series of pop-up windows. When specification is complete, DECKBUILD automatically produces a syntactically correct input deck. You can edit the decks at any time. Multiple decks are produced if input parameters are looped, and parameters can be extracted from the calculated results. Multiple simulators can be called from within a single input deck, and information transfer between simulators is transparent.

DECKBUILD allows precise user control of how an input deck is run, with stop at, pause, restart and single step capabilities. It provides a history function that permits you to backtrack to a previous point in the deck and then continue computation at this point. This capability is extremely useful for interactively developing a simulated process flow specification. DECKBUILD can invoke other simulator support tools, such as TONYPLOT, DEVEDIT, and MASKVIEW. DECKBUILD also works with the OPTIMIZER. Optimization targets may include structural dimensions, device parameters after a complicated electrical test, and any intermediate outputs.

DECKBUILD was designed to minimize the time required to build up an input deck and to calibrate that input deck to first order accuracy. An input deck may be built up in stages, a line at a time if required. A typical sequence of actions in the construction of an input deck might be as follows:

1. Write Deck Fragment
2. Simulate Fragment
3. Plot/Check structure
4. Write second input deck fragment
5. Simulate this second fragment
6. Plot/Check structure?
7. Find a mistake!
8. Re-initialize to the end of first fragment in the input deck

9. Adjust the second input deck fragment
10. Resimulate the second fragment
11. Plot/Check structure.... OK now.
12. Write a Third Input Deck Fragment

This is a “two steps forwards, one step backwards” approach to the building of an input deck. You can also use the built-in OPTIMIZER to adjust values into line with known measured values. The DECKBUILD OPTIMIZER is designed to be used in the same environment, and is designed to be used as a step by step calibration tool.

Once an input deck is complete, you can re-run it with “look see” variations or load it into the VWF database, where automated DOE split lots and RSM model generation may be performed.

1.2.2: TonyPlot

The Visualization Tool provides comprehensive interactive scientific visualization capabilities. All of the usual ways of displaying scientific data are supported. These include xy plots with linear and logarithmic axes, surface and contour plots, Smith charts, and polar plots. Virtually every characteristic of the plots, including the text and position of labels, can be specified. Full hardcopy capabilities are also supported. TONYPLOT includes animation features that permits viewing a sequence of plots in a manner showing solutions as a function of some parameter. The parameter can be varied under slider control or frames can be looped continuously, a feature helpful in developing physical insight. TONYPLOT supports production of hard copy plots on a wide range of printers.

1.2.3: DevEdit

The Structure Editor is an interactive tool for specifying and modifying structures. It includes a meshing module that supports mesh generation, refinement, and unrefinement. You can define and modify doping using analytical functions. DEVEDIT can be used stand-alone, or can be invoked by DECKBUILD. Large devices with many grid points can be specified completely using DEVEDIT, making this tool valuable as a pre-processor for 2D device simulators. A special mode of DEVEDIT supports the definition and meshing of 3D structures.

1.2.4: MaskViews

The IC Layout Editor is a versatile IC layout editor used to specify layout information to process simulators. MASKVIEWS supports simulation based experimentation with layout variations. Experimentation based simulation was previously restricted to the varying of process flow parameters only. MASKVIEWS supports experimentation dealing with phase shift masking technologies, critical dimensions, misalignment tolerances and global shrinks. It is fully interfaced to GDS2 Stream formats so that complete IC layouts can be imported and exported. Small subregions can be selected for detailed analysis. You can use MASKVIEWS interactively through DECKBUILD.

1.2.5: Optimizer

The OPTIMIZER provides optimization capabilities that are sophisticated, efficient, and easy to use. It was originally developed for use with the UTMOST parameter extraction and device modeling software. Optimization capabilities are valuable for technology design and for calibrating and tuning model coefficients and input parameters used by simulators.

Control of the OPTIMIZER is integrated into DECKBUILD. The use of the OPTIMIZER under the control of DECKBUILD can extend across multiple simulators. In other words, you can tune input parameters of process simulators to produce specified electrical characteristics from a device simulator.

1.2.6: Manager

MANAGER is a simple application manager that supports point-and-shoot and drag-and-drop use of a variety of files and the support tools. The use of this tool is very intuitive. For example, you can select a structure file using the mouse and drag it to the TONYPLOT icon to plot the file on the screen.

1.2.7: SPDB

The SEMICONDUCTOR PROCESS DATA BASE (SPDB) is a separate product, not an INTERACTIVE TOOL. But, it can be used with DECKBUILD in a fully integrated manner. Refer to the SPDB USER'S MANUAL for additional information.

1.3: VWF Automation Tools

1.3.1: The Virtual Wafer FAB

The systematic development of semiconductor technologies involves two stages. First, relationships between input variables (i.e., processing parameters) and output responses (i.e., electrical behavior) are established. Second, this information is then used to obtain optimum responses.

Examples of input variables are implant doses and energies, diffusion times and temperatures, and mask layout variations. Taking CMOS technologies as a specific example, the responses of interest include threshold voltages, subthreshold slopes, transconductances, leakage currents and breakdown voltages.

This procedure is simple in principle but difficult in practice. There are numerous input variables and output responses. A lot of effort is required to establish the relationships between process parameters and electrical characteristics. It is also not easy to define figures of merit associated with optimum responses. The relationships between inputs and responses are usually investigated by running split lots in a wafer fab and measuring the electrical characteristics of special test structures. This is expensive and time consuming, even when modern techniques are used to design experiments and analyze the results. The costs are escalating rapidly as wafer sizes increase and processing equipment becomes more expensive.

Using physically-based process and device simulation may be quicker and cheaper. But to use simulators, engineers have had to acquire specialized simulation-related knowledge. Even when simulation skills have been acquired, performing significant studies that require many runs has been time consuming and inconvenient. You had to design the study, generate all the input decks required, submit all the runs, store all the results, and only then analyze the results. The VWF AUTOMATION TOOLS eliminate these problems.

The VWF AUTOMATION TOOLS contain software tools that design simulation based experiments, generate input decks, submit runs in a networked computing environment, store results, and analyze results. These capabilities make it extremely easy for engineers to perform large simulation-based experiments that mirror existing experimental development procedures. The VWF AUTOMATION TOOLS use the tools of the VWF INTERACTIVE TOOLS. For example, DECKBUILD, TONYPLOT, DEVEDIT, MASKVIEWS, and OPTIMIZER as well as many additional specialized tools. Data is automatically and transparently transferred between all the tools that comprise the VWF.

1.3.2: VWF Experimentation

Experimentation in the VWF AUTOMATION TOOLS starts with a baseline input deck. The effects of varying specified input parameters is investigated. An automated sensitivity analysis of the effect of input parameter variations on outputs is usually performed first. Splits on any desired parameters are then performed. You can specify a few parameter variations or set up a large split lot type experiment. You can define the values of split parameters. These values can also be generated automatically using built-in design of experiments (DOE) capabilities. A graphical worksheet associated with an experiment is generated automatically. You can directly make changes to the experiment on the worksheet. The input decks associated with an experiment are generated and submitted automatically. The results are collected and stored in a database. The data is then conveniently analyzed using response surface methods and graphical animation.

1.3.3: Advanced Features

The VWF AUTOMATION TOOLS contain numerous advanced features, many of which represent significant advances in semiconductor technology CAD. Some of these features are summarized below.

VWF Database

The heart of the VWF AUTOMATION TOOLS is an object-oriented database that holds all input and output data in an efficient manner. You access the database through an intuitive icon and menu-driven GUI. The status of an experiment in progress can be displayed graphically and can be modified.

The database supports libraries and workspaces. Libraries are used to store relatively long-lived, stable objects that are useful across a project or organization. Workspaces typically contain shorter-lived information used by an individual engineer. Objects can include working input decks, process flows, mask layouts, and electrical tests.

DECKBUILD can be used as a hierarchical folding editor when it is used in the VWF AUTOMATION TOOLS. Whole sections of text can be replaced by labels indicating their function, and each section of text can be stored in the database as an object. Full decks can be constructed by using the mouse to drag-and-drop objects to the required position in DECKBUILD.

Process Flow Editor

Decks can also be built by dropping objects directly onto a flow editor. This allows a full process flow to be built up visually in terms of graphical icons. Any number of layout cross-sections and device tests can be added to the process flow at any time. The ability to experiment with layout variations in the same way as process variations is a unique and particularly valuable aspect of the VWF AUTOMATION TOOLS.

Sensitivity Analysis

Sensitivity analysis on output values with respect to input parameters is readily performed. Parameters are ranked in order of their influence on the output, and the effect that each parameter has on the final results is listed. A special feature on the split point editor allows a sensitivity design tree to be created automatically without any further user interaction.

Split Points, Experimental Trees, and Worksheets

Any statement parameter in the input deck can be defined as an experimental split point. An arbitrary number of different values can be specified for each selected parameter, resulting in a tree of experiments. A graphical worksheet is generated automatically. This worksheet allows you to view input parameters and output results. The worksheet supports filtering of values and allows columns to be specified as functions of other columns. Input decks are automatically generated for each cell in the worksheet. You can add additional experiments at any time by pointing to the parameter to be varied and specify the additional values. Simulation results are logged automatically to the worksheet, which is updated as simulations are completed.

Experimental Design

Parameter values for splits can be generated automatically through the use of experimental design techniques. Full and partial factorial, random, Latin Hypercube, and composite designs are presently supported.

Intelligent Execution

The experimental tree structure is exploited to automatically minimize the total required computation. Intermediate results are re-used for child nodes, thereby eliminating unnecessary duplication. Filter limits can be specified so that branches of a tree are eliminated if intermediate extracted parameter values go outside acceptable ranges.

Networked Execution

Jobs are generated and submitted automatically. All the computers in a network are used and subject to various conditions that you can set. These conditions include run priorities and available times and days for job submission to each workstation. Continuous graphical displays of the status of the network and of experiments in progress are available. Automated job submission allows all of a company's networked computers to be used in parallel for simulations, and exploits previously idle time, such as overnight and weekends.

Response Surface Analysis

Once an experiment has been run, the results can be conveniently analyzed using built-in regression techniques. Formulae that express output responses as functions of the input parameters are produced. Second order polynomial surfaces are currently calculated.

Integrated Graphics

Interactive 2D, 3D and 4D (color contoured 3D) graphics make it easy to view and analyze results. Scatter, schmoos, residual, contour and surface plots are available. There is feedback between the scatter plots and the worksheet displays. Therefore, scatter points on the graphics plot are highlighted as the mouse pointer crosses the associated row in the worksheet.

Presentation Quality Printouts

You can print all of the worksheet, regression and graphical displays. Pagination, formatting, titling and labeling are performed in a way that allows printouts to be used directly in reports and presentations.

On-Line Help

The VWF INTERACTIVE TOOLS all include on-line manuals. These provide instant access to help with tutorials on use the tools. The on-line help is not purely text based: it includes many graphical images and diagrams that clarify operation and use issues.

1.3.4: Productivity Enhancements

The VWF AUTOMATION TOOLS increases productivity in ways that are both direct and indirect. Many engineers who previously avoided simulation have started to use it. Other engineers use it more intensely now that it is very convenient to perform large simulation based experiments. Computer hardware is used more efficiently, as the VWF AUTOMATION TOOLS submits jobs in parallel using an entire network, including overnight and weekends.

The knowledge of experienced engineers disseminates through an organization, in the form of easily manipulated library objects that contain tested descriptions of process descriptions, and electrical tests. Productivity is also enhanced by the encapsulation of state-of-the-art techniques for experimental design and data analysis.

The bottom-line benefit of the VWF AUTOMATION TOOLS is that it helps substitute simulation, which is relatively quick and cheap, for experimental split lots, which are time consuming and expensive.

2.1: Overview

This tutorial shows you how to use DECKBUILD and TONYPLOT to perform effective process and device simulation on either a UNIX or Linux platform.

The VWF INTERACTIVE TOOLS make it easy to access the capabilities of the ATHENA process simulator and the ATLAS device simulator. The VWF INTERACTIVE TOOLS and their functions are as follows:

- DECKBUILD — interactive run time environment
- TONYPLOT — scientific visualization
- DEVEDIT — structure specification and meshing
- MASKVIEWS — IC Layout editor and interface
- OPTIMIZER — black-box optimizer

Two of these tools, DECKBUILD and TONYPLOT, are intended for use by all users of ATHENA, ATLAS, MERCURY, CLEVER and other SILVACO simulators.

DECKBUILD provides convenient ways to specify problems, run simulations, switch between simulators, extract parameters from simulation results, and invoke other VWF INTERACTIVE TOOLS. DECKBUILD is the “command center” of SILVACO’s interactive simulation environment.

TONYPLOT is a powerful and versatile scientific visualization tool that plots the data produced by process and device simulators. TONYPLOT can be invoked by users or by DECKBUILD.

This tutorial illustrates the use of DECKBUILD and TONYPLOT in a typical application. The tutorial is built around the process and device simulation of an MOS transistor. Some familiarity with the basic concepts of silicon device processing and MOS transistor operation is helpful, but not essential, for working through the tutorial.

Note: The VWF Interactive Tools for PC Windows are different. For more information, see the PC INTERACTIVE TOOLS USER’S MANUAL.

2.2: Introduction

The core tools of semiconductor technology CAD include process simulation (ATHENA), device simulation (ATLAS), circuit simulation (SMARTSPICE), device characterization and modeling (UTMOST), and other SILVACO simulators. These tools can be used in a stand-alone “batch” mode, or from within the DECKBUILD interactive environment. Running the core tools from within DECKBUILD provides many advantages, including ease of use, convenience, and access to additional capabilities.

The tutorial will show you how to do the following:

- Start DECKBUILD and load an example input deck.
- Run ATHENA under the control of DECKBUILD, to simulate a simple LDD MOS process.
- Use TONYPLOT to view the results generated by ATHENA.
- Run ATLAS under the control of DECKBUILD, to simulate the MOS structure generated by ATHENA.
- Use TONYPLOT to view the results generated by ATLAS.
- Use DECKBUILD to extract device parameters from results calculated by ATLAS.

The tutorial does not attempt to describe the detailed use of the ATHENA and ATLAS simulators. The user manuals for ATHENA and ATLAS document the features and capabilities of these tools, and include chapters that help new users get started.

Before you can work through this tutorial you must have ATHENA, ATLAS, DECKBUILD, and TONYPLOT installed on your system. If you are unsure about the status of your software installation, see the SILVACO INSTALLATION GUIDE, or ask your local system administrator for assistance. The principles of using DECKBUILD and TONYPLOT, shown in this tutorial, are valid for users of other SILVACO simulators.

2.3: Getting Started

To start DECKBUILD, enter

```
deckbuild &
```

at the system prompt, and press the return key.

It may take several seconds for DECKBUILD to load. If DECKBUILD does not appear, or if you receive error messages at this point, ask your systems administrator to check the software installation or license.

When DECKBUILD starts, a DeckBuild Application Window appears on your screen that looks similar to Figure 2-1 (version numbers and directory names vary depending on your installation).

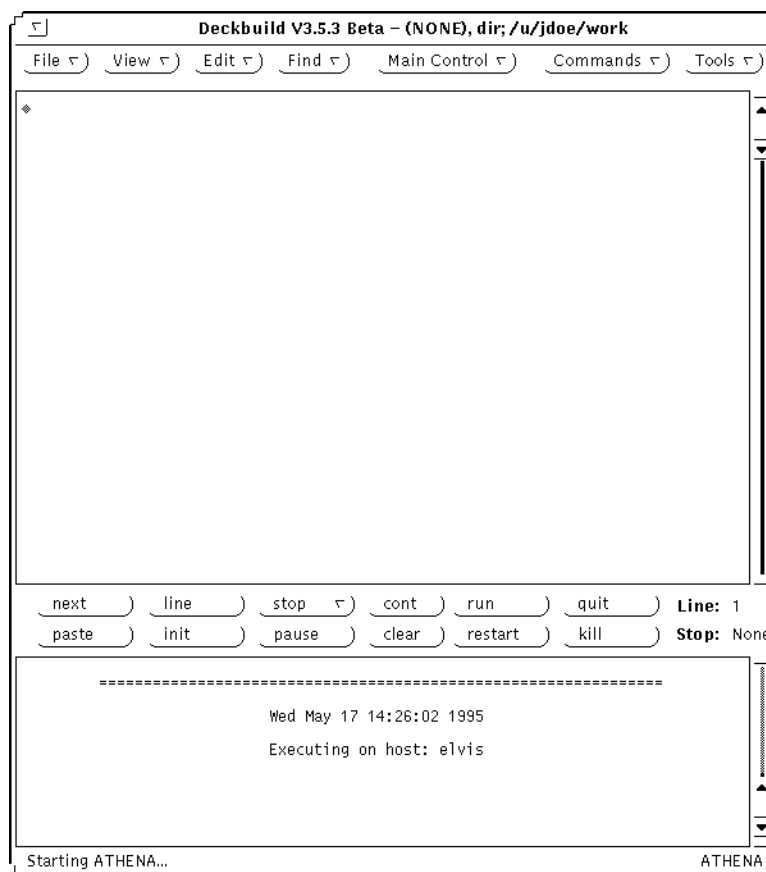


Figure 2-1: DeckBuild Main Window

The DECKBUILD application window contains:

- An upper text edit region that will hold simulator input commands.
- A lower ‘tty’ (scrolling teletype) region that will display run time output from the simulators.
- A set of file control buttons located at the top of the text edit region.
- A set of simulator control buttons located between the text edit region and the tty region

The name of the simulator that is currently running under the control of DECKBUILD is displayed at the lower right hand corner, in the footer bar of the DECKBUILD application window. Status messages appear in the lower left hand corner of the footer bar.

When DECKBUILD loads, ATHENA starts by default. You can change this default for a particular session by specifying a different simulator as a parameter of the `deckbuild` command. You can also change the local default by specifying a different simulator in the **Main Control** window of DECKBUILD, and then saving this as the default. CHAPTER 4: “DECKBUILD” discusses DECKBUILD thoroughly.

The first text that appears in the tty region is ATHENA start up information. This consists of a title banner and a summary of ATHENA products licensed for use with your installation. This information will be followed by the ATHENA prompt:

```
ATHENA>
```

The prompt indicates that ATHENA is ready to execute instructions.

The input commands that you want ATHENA to process must be entered into the text edit region. The commands are then passed to the lower tty region for execution in a way that you specify using the simulator control buttons.

A set of input commands can be defined in several ways, including:

- Typing commands directly into the text edit region. The statements must be written using the appropriate simulator syntax, which is documented in the user manual of the simulator.
- Filling in popup windows that are invoked from the DECKBUILD **Commands** menu, and having DECKBUILD write syntactically correct statements.
- Loading and editing an existing input file.

The first method is sometimes preferred by experts who are familiar with simulator input syntax. The second method is often preferred by new users, since they need only specify problem-oriented information, without knowing the simulator syntax. The third method (incremental modification of an input deck that is known to work) is often an effective way to get started on a new problem.

DECKBUILD comes with a large number of examples that provide a starting point for the third mode of operation. This tutorial uses an application example that is supplied with DECKBUILD. Please see Chapter 4: “DeckBuild” and the ATHENA and ATLAS USER’S MANUALS (or whatever simulator you wish to use), for information about the other modes of specifying input.

2.4: Selecting and Loading An Example

Do the following to bring up a list of the examples that are supplied with DECKBUILD.

1. Move the cursor over the **Main Control** button, and click and hold the right hand mouse button. The Main Control menu will appear. One of the options on this menu is labeled **Examples...** as shown in Figure 2-2.

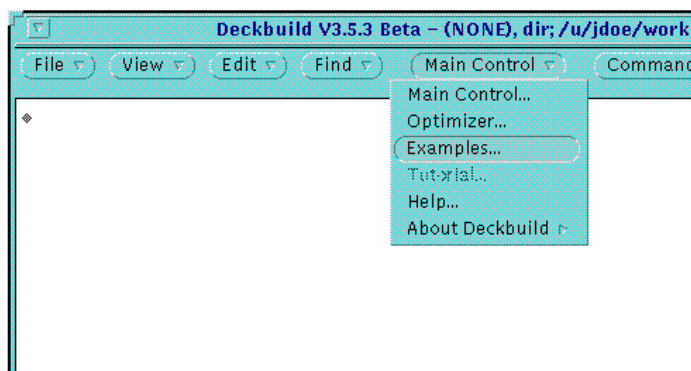


Figure 2-2: Main Control Menu

2. Without releasing the right hand mouse button, move the cursor down to select the **Examples...** item. Then, release the mouse button. A window listing a number of example categories will appear (Figure 2-3).

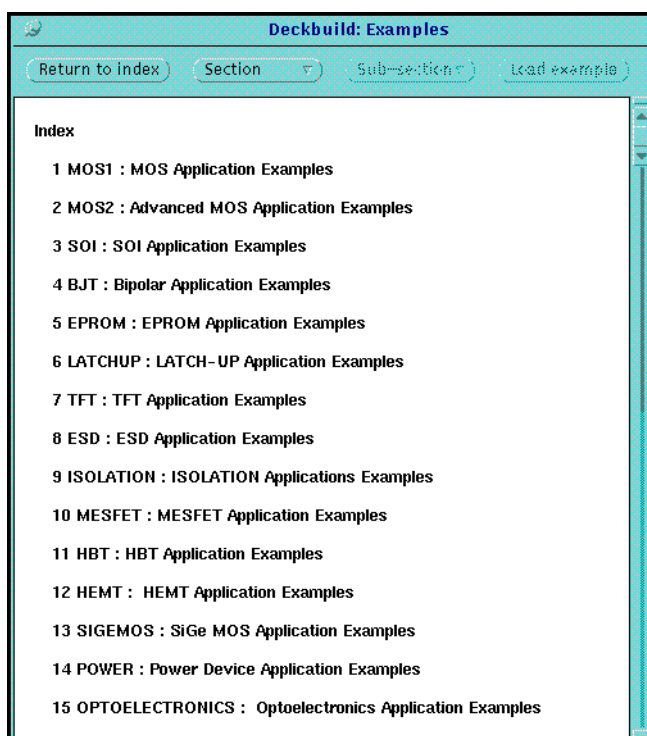


Figure 2-3: Main Control Menu

This tutorial uses an example from the MOS1 category. Select the MOS1 category by double-clicking on this line of text, or by selecting MOS1 from the **Section** menu.

The tutorial uses the second example in the MOS1 category. This has the title **NMOS: Id/Vgs and Threshold Voltage Extraction** and uses an associated input file named `moslex01.in`. Select this example by double-clicking on the line numbered **1.1** or by using the **Sub-Section** menu. Text describing this example will be displayed. Take a few moments to read the description.

It is very easy to print a copy of the examples documentation. Click on the right hand button in the window where the description is displayed. A **Print** menu appears, offering you the choice of printing a page, a section, or the complete document. Click on your choice. Please consider carefully before printing the complete document as this is a lot of information.

Note: The TCAD EXAMPLES USER'S MANUAL VOLS. I-III have descriptions of all examples.

Select the **Load Example** button to load the input deck associated with this example into the DECKBUILD text edit region. This action also causes other files related to this example to be copied into your current working directory. Click on the push-pin to dismiss the Examples window.

The text edit region now contains all the commands required to perform process simulation, extract information of interest from the simulated results, view the simulated results, perform device simulation and extract device parameters from the simulated results.

Take a moment to scroll down through the text in the text edit region. There are a series of ATHENA statements, followed by other statements including DECKBUILD extract statements, TONYPLOT statements, a `go atlas` statement, and some ATLAS statements.

2.5: Process Simulation

The first portion of the input specifies process simulation of an LDD MOS transistor. Now, start running this portion of the input deck.

This section of the tutorial describes the following activities:

- Running a simulation
- Displaying a structure
- Using the history function
- Creating two structure files for comparison
- Overlaying two plots
- Using TONYPLOT to produce 2-D plots
- Generating interactive cutline plots
- Extracting some process parameters

2.5.1: Running a Simulation

The runtime control buttons (Figure 2-4) are located between the DECKBUILD text edit and tty regions. These buttons are used to control the way a simulation runs interactively.

You can specify the simulation will run in any of the following ways:

- Run one step at a time (i.e., with full interactive simulation control).
- Run to a specified stop point.
- Run the entire input deck.

Simulators can run under batch control or optimizer control. These methods of execution are not covered in this tutorial.

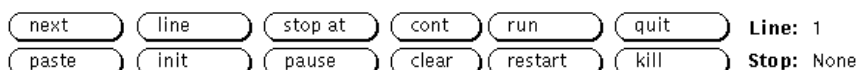


Figure 2-4: Runtime Control Buttons

Stepping Through A Simulation

We will run the first part of the process simulation by going through one command at a time. This single-step mode is often used for identifying errors, and during tuning and calibration.

To execute commands one step at a time, select the **next** button from the DECKBUILD control panel. The **next** command in the text edit region is sent to the active simulator, and the cursor in the text edit region moves down by one line. The previous command appears at the simulator prompt in the tty region. It is processed by the simulator and any associated output is displayed in the tty region. The command that is currently executing is displayed in reverse video in the text edit region. The line number of the current command is shown on the right hand side of the control panel after the **Line** label.

If you click on the **next** button several times in a short period of time, the cursor will move ahead of the simulator as the commands take some time to execute. The simulator continues to process commands until it catches up to the cursor. The simulator then waits for the next command to be sent.

Single-step through the simulation until you complete the gate oxidation step (line 46). Notice the different commands sent to ATHENA and the run time output that appears in response to these commands.

The gate oxidation is followed by an extraction step that determines the oxide thickness:

```
extract name="gateox" thickness oxide mat.occno=1 x.val=0.05
```

This command is an example of DECKBUILD's powerful EXTRACT capability, which allows you to determine various features of the device that is being simulated. This statement determines the gate oxide thickness. More complicated examples of the EXTRACT capability are used Section 2.5.11: "Extracting Process Parameters" and also in Chapter 5: "DeckBuild:Extract".

Single step through the extract statement. Note that the prompt in the tty region changes from ATHENA to EXTRACT as actions associated with this command are performed. The prompt reverts to ATHENA when the EXTRACT operation is complete.

2.5.2: Plotting A Structure

At any time during process simulation, TONYPLOT can be used to display the current structure. DECKBUILD makes it very easy to do this without any modification of the input deck.

Before attempting to plot the current structure, you should click anywhere in the text edit region to deselect any text selected (i.e., as the result of executing an instruction). This is necessary because TONYPLOT attempts to interpret any text that is selected (i.e. highlighted in reverse video) as the name of a file to be loaded.

Having made sure that there is no selected text, you should now position the cursor over DECKBUILD's **Tools** button. This is the righthand most file control button. Click and hold the left mouse button and a drop-down menu appears. Then, move the cursor down to **Plot** and move the cursor to the right until a sub-menu appears (Figure 2-5). Select **Plot Structure...** and release the left mouse button.

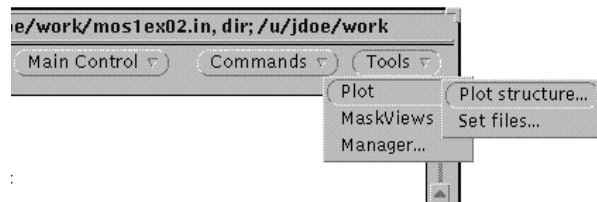


Figure 2-5: The Tools Menu

This will start TONYPLOT, which loads and displays the current structure. This may take several seconds. DECKBUILD announces that it is starting TONYPLOT by displaying the status message: "Plotting ..." on the left hand side of the footer bar below the tty region.

When TONYPLOT starts, it displays a **Welcome** window. Select **OK** on the window to display the current structure.

2.5.3: Using TonyPlot

TONYPLOT displays the materials of the various layers with a 'SIMS'-like doping profile superimposed. The plot will look similar to Figure 2-6.

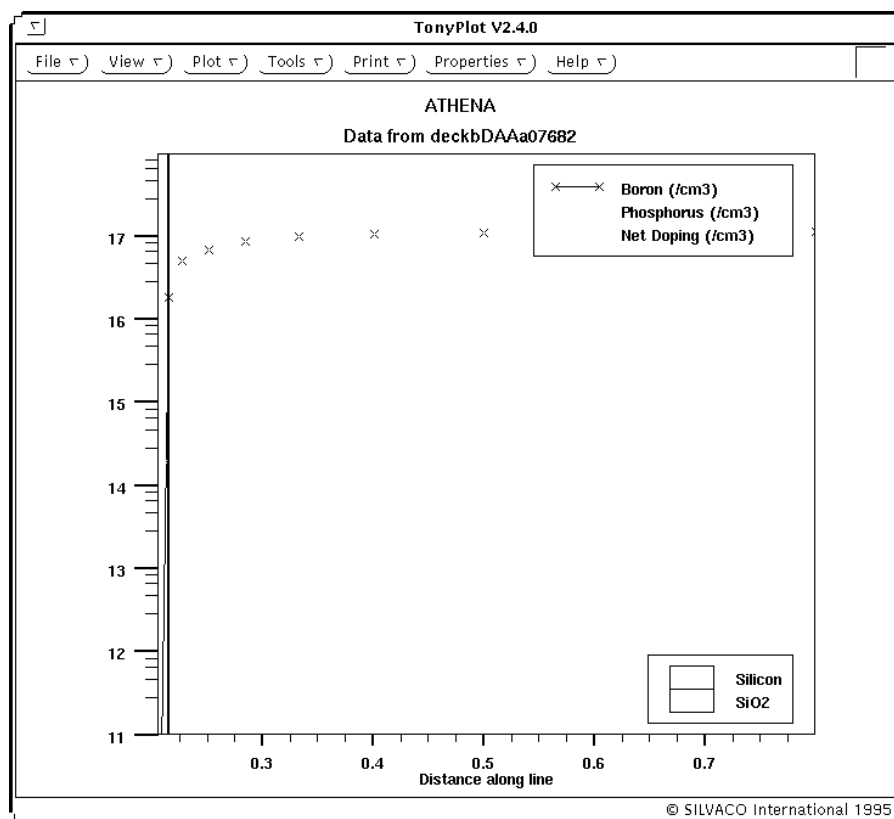


Figure 2-6: The Doping Profile

These results are displayed in 1-D when ATHENA is a 2-D process simulator. The reason is that ATHENA performs calculations in 1-D until an instruction causes the structure to become non-planar. From this point on, it uses the full 2-D mode. This automatic mode switching saves CPU time. TONYPLOT automatically detects when a structure is the result of calculations performed in 1-D mode and it displays the results accordingly.

Modifying The Appearance Of The Plot

You can use the buttons along the top of the TONYPLOT window control to change the appearance of the plots that are generated. To start modifying the present plot, select the **Display...** item of the **Plot** menu (see Figure 2-7).

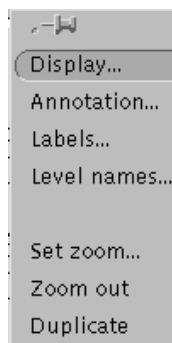


Figure 2-7: The Plot Menu

This causes the **Display** window shown in Figure 2-8 to appear.

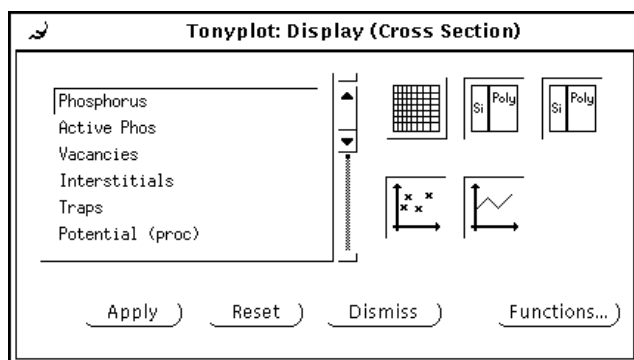


Figure 2-8: Display Cross-Section Window

This window contains several options that control the appearance of the graph, including:

- Which quantities are plotted. The available options are displayed in a scrolling list. To plot a particular quantity, simply select it from the list. The **Phosphorus** species is selected in the figure above.
- Whether the simulation grid is overlaid on the graph.
- Whether the interface between different regions and materials are delineated using lines or different colors or both for different regions.
- Whether the graphs have symbols displayed at the data points or have the data points connected or both.

Changes defined in the display window only affect the displayed plot after the **Apply** button is selected. The **Reset** button resets the controls to the state they had when the display window was first displayed.

Experiment with making changes to the displayed plot. Use the **Dismiss** button to remove the **Display** window when you have finished experimenting.

Zooming And Panning

You can inspect the regions of the graph in more detail by zooming in on the area of interest. To zoom in, select a rectangular area of the graph by using the mouse. Using the left mouse button, select the lower left corner of the region of interest. Keeping the mouse button depressed, move the pointer to the upper right corner of the region of interest, and release the button. The graph is replotted, showing only the newly defined region of interest.

Figure 2-9 shows the details of the doping concentrations in the oxide.

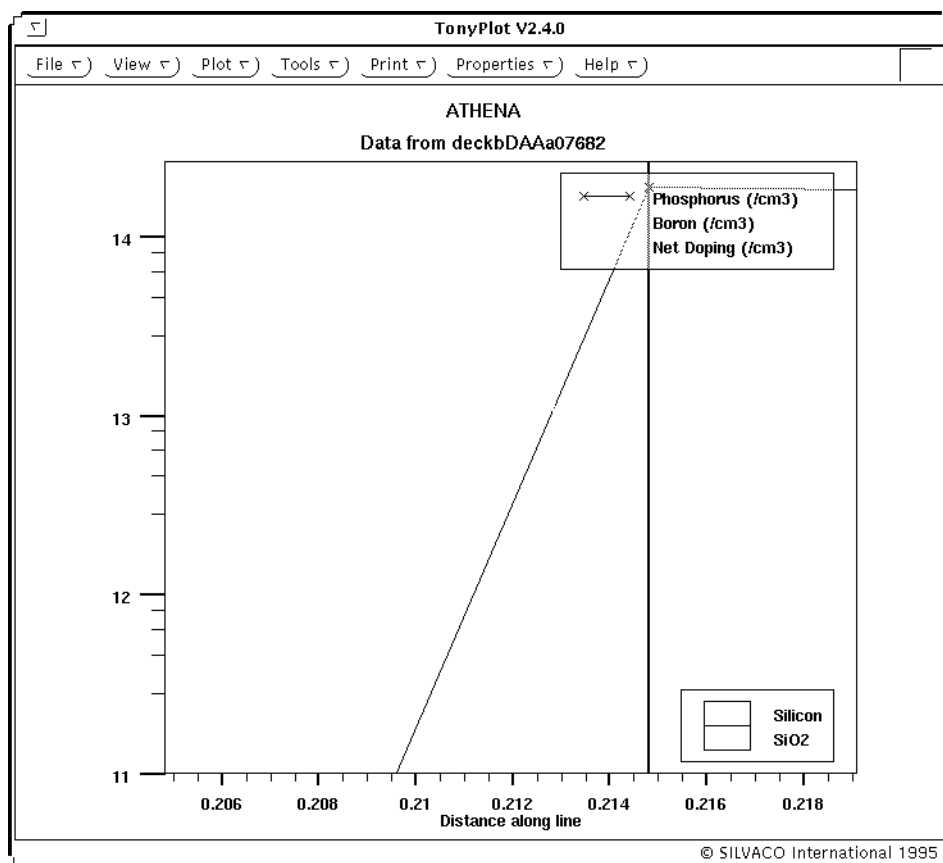


Figure 2-9: Doping Concentrations in the Oxide

After you zoom into a region of a graph, a zoom/pan box appears in the upper left corner of the window, as shown in Figure 2-9. Selecting any of the eight direction arrows on this box pans the graph in that direction. In other words, plot the graph in areas adjacent to the current area in the given direction. To return to the original, (un-zoomed) graph select the button that is marked with a diamond in the center of the zoom/pan box.

Printing A Plot

Select the **Print** button to print out the plots. This sends a copy of the plot directly to your default printer. You can use other printers with the **print** button.

Note: Your system manager will be able to tell you which is your default printer and where it is located, if you are not sure about this.

2.5.4: Using The History Function

The history function provided by DECKBUILD allows you to make corrections and changes during an interactive simulation session without requiring a re-simulation of the entire input deck from the beginning. This can save a lot of time during input deck debugging, and when performing verification or calibration.

The history function allows you to move backwards through an input deck by selecting a previously simulated command line and initializing the simulator back to that point. The history function works in the following way. After each “meaningful” process simulation step (i.e., when a change to the structure has occurred), a simulator state file will be automatically saved in the current working directory.

The history files are named automatically as `.historynn.str` where `nn` is a sequence number. The history files are stored as Standard Structure files. As ATHENA was executing, you may have noticed the commands to save these files at the ATHENA prompt. For example:

```
struct outfile=.history09.str
```

DECKBUILD remembers which files are associated with each command, as long as commands have not been not added, deleted or changed since the history files were created. You can also highlight the history structure file name, and click on **Tools** in DECKBUILD, which starts TONYPLOT and displays the history file.

To re-initialize the simulator to the previously simulated Well Drive step, double or triple-click on the diffusion line for the Well Drive so that it is highlighted as shown in Figure 2-10.

```
..
#N-well implant not shown -
#
# welldrive starts here
diffus time=50 temp=1000 t.rate=4.000 dryo2 press=0.10 hcl=3
#
diffus time=220 temp=1200 nitro press=1
#
diffus time=90 temp=1200 t.rate=-4.444 nitro press=1
```

Figure 2-10: Reinitializing the Simulator

With at least some of the text on this line highlighted, select the **init** button on the DECKBUILD control panel. This resets the position of the current simulation line to the end of the Well Drive Diffusion step. You can now make corrections or modifications at any point downstream, and re-simulate.

As an example of a downstream change, we will change the Well Drive time from 220 to 200. The edited command should look like this:

```
diffus time=200 temp=1200 nitro press=1
```

Use the next button to step through the modified process flow.

To continue with this tutorial, proceed with the simulation until you have again extracted the gate oxide thickness.

Note: Meaningful process simulation steps are steps that change either the topography or the dopant concentrations of the structure. For example, implant and etch. Non-meaningful steps are steps that simply interrogate the simulator for information. For example, writing structure file or extracting a parameter.

Explicitly Saving State

The history files are very useful during an interactive session. But, they may become invalid due to changes to the input deck. Also, they do not have descriptive names that make them easy to associate with particular points in the process flow.

You can avoid these problems by saving the state of the simulation at any point, to a file whose name you specify. A structure file named `bpsg_dep.str` is generated by the command:

```
struct outfile=bpsg_dep.str
```

Note: Standard Structure files must have a `.str` extension.

It is a good idea to perform a user-defined save immediately following major process steps that are time consuming to simulate, such as two-dimensional diffusions or Monte Carlo based calculations.

The **init** button also works with structure files that you saved. To initialize on a named Standard Structure file, select the name of the saved structure file in the text edit window. Figure 2-11 shows the file `vtadjust.str` selected in the text edit window.

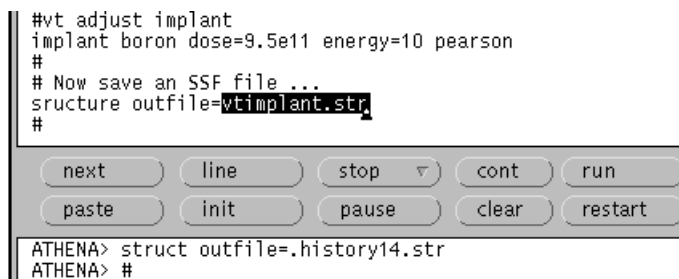


Figure 2-11: Selecting a Structure File in Deckbuild

Selecting the **init** button re-initializes the simulator to the point where the selected file was saved. The current point of execution in the input file is set to the point directly after the statement that saved the selected file. The line

```
init infile=vtadjust.str
```

which appears in the tty region is the command executed when the **init** button is selected.

2.5.5: Comparing Structure Files

You can use TONYPLOT to compare two structures from different points in the process flow. To illustrate this, compare the structures that resulted just before and just after the Vt adjust implant step.

To perform this comparison, first generate a structure file for each structure. Then, modify the input deck to generate the necessary structure files, and use the history mechanism to initialize to a point just before the gate oxidation step. In this way, we avoid having to re-simulate the complete input deck.

To re-initialize to the point of the sacrificial oxide strip, select the text shown in Figure 2-12 and press **init**.

```
#sacrificial "cleaning" oxide
diffus time=20 temp=1000 dryo2 press=1 hcl=3
#
etch oxide all
#
#gate oxide grown here:-
diffus time=11 temp=925 dryo2 press=1.00 hcl=3
```

next line stop ▾ cont run

paste init pause clear restart

ATHENA> struct outfile=.history18.str

Figure 2-12: Reinitializing the Etch Statement

You are now free to modify the input deck anywhere subsequent to this point. Add the following two statements to save two Standard Structure files, one before and one after the Vt adjust implant step:

```
struct outf=gateoxide.str
struct outf=vimplant.str
```

These lines are added by typing directly into the text edit window at the appropriate points as indicated in Figure 2-13.

```
#gate oxide grown here:-
diffus time=11 temp=925 dryo2 press=1.00 hcl=3
#
# Save the structure after oxide growth
struct outf=gateoxide.str
#
# Extract a design parameter ...
extract name="gateox" thickness oxide mat.occn=1 x.val=0.05
#
#vt adjust implant
implant boron dose=9.5e11 energy=10 pearson
#
# Save the structure after Vt implant
struct outf=vimplant.str
#
```

next line stop ▾ cont run

paste init pause clear restart

Figure 2-13: Adding Lines in Deckbuild's Textedit Window

Single-step until the command `struct outf=vimplant.str` is executed. At this point, the two Standard Structure files, with the names `gateoxide.str` and `vimplant.str`, have been saved to your current working directory. These structure files are loaded into TONYPLOT and overlayed for comparison.

2.5.6: Overlaying Two Plots

TONYPLOT allows you to load up to 128 plots into a single session. Any of these plots can be overlaid simply.

To load a file into TONYPLOT, select the **File→Load Structure....** This creates a menu of possible files that can be loaded (Figure 2-14).

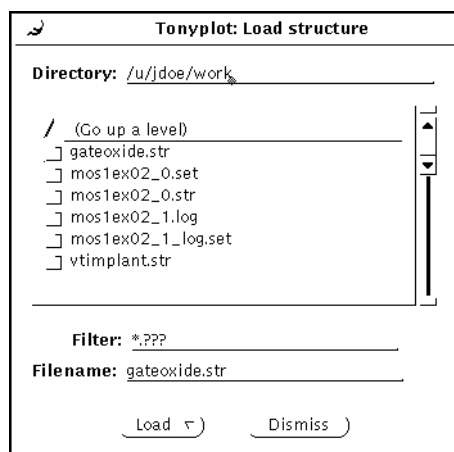


Figure 2-14: The Load Structure Window

The files `gateoxide.str` and `vtimplant.str` should exist in your current working directory and should appear on the list of files that can be loaded into TONYPLOT.

Any plot shown on the TONYPLOT screen can be selected. When you select a plot, it is surrounded with a band. Plot selection and deselection is controlled by clicking the center (adjust) mouse button on a plot. Select and deselect some plots to become familiar with this process.

To overlay two plots:

1. Select both plots so that both are surrounded by the band/border.
2. Select in TONYPLOT **View→Make Overlay**.

A third plot now appears as an overlay. This new plot shows the profiles from before and after the Vt adjust implant overlaid on a single plot.

2.5.7: Continuing the Process Simulation

You will now complete the process simulation portion of this tutorial. After the polysilicon etch, the structure becomes non-planar, ATHENA switches automatically to 2-D calculation mode, resulting in increased CPU requirements.

Rather than single-stepping using the **next** button, use the **stop** function to perform simulation up to a selected point of interest.

The 'Stop At' Function

A stop point defines a position in the flow of commands where the simulator will stop executing and await your next action. When you define **stop** point, selecting the **run** button or the **cont** button causes the simulator to execute commands as far as the stop point and then waits.

To set a stop point, position the cursor in the DECKBUILD text edit region on the input line where you want the simulation to stop, and select a few characters so that they are highlighted. Then, select the **stop** button from the simulator control panel. This sets a stop and the **Stop:line** display will be updated to show the line number of the stop. You can clear a set stop point at any time by selecting the **clear** button.

After setting a stop point, continue the simulation by selecting the **cont** button on the simulator control panel. DECKBUILD now sends commands to ATHENA, up to the preset stop point, and then pauses the simulation directly before this point. As the simulator is running, take some time to observe the ATHENA commands being executed and the run time output that is being generated.

Click on the **next** button to make sure that the simulator runs to include the metal etch step directly after the defined stop point. The polysilicon etch makes the structure non-planar. The simulator has to do more work in 2-D mode, and progress through the commands is slower.

Most of the commands being used are straightforward and their meaning is clear to most process engineers. The following lines may require some explanation:

```
depo poly thick=0.2 divi=10
```

This command specifies that a layer of polysilicon 0.2 microns thick is deposited and that 10 grid layers is defined.

```
etch poly left p1.x=0.35
```

This command defines the position in the x direction from which the polysilicon to the left is removed. This command defines the length of half of the transistor, which is reflected about the right axis later on to make the full device.

```
method fermi compress
```

This command switches on some physical models for subsequent oxidation and diffusion steps. All following steps use these models. These models are in fact the ATHENA defaults, and are only defined here as an illustration of the `method` command. The ATHENA USER'S MANUAL provides full documentation of all available model.

After a few minutes, the simulation will reach the point defined by your most recent selection of the **next** button. The DECKBUILD window should then look like Figure 2-15.

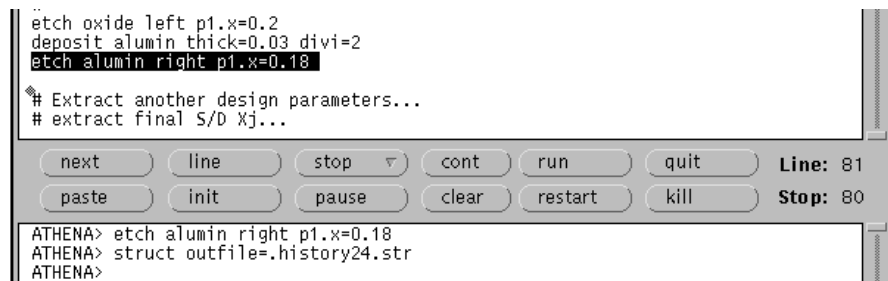


Figure 2-15: The Deckbuild Window Prior to Plotting

2.5.8: Plotting 2-D Structures

Now plot the current structure in TONYPLOT using the same procedure from the previous section. In other words, deselect any selected text and select **Tools→Plot structure**.

By default when TONYPLOT starts up, it displays the 2-D material structure of the current structure. To examine the topography of the structure in more detail, zoom to different areas of the device.

TONYPLOT supports two methods for viewing the impurity concentrations (or other solution variables) in two dimensional structures. These are two dimensional contour plots (the default when TONYPLOT is invoked for a two dimensional structure) and one dimensional profiles along defined cutlines. For this tutorial, a contour plot of the net doping will be obtained first, and then some plots along cutlines will be defined.

2.5.9: Contour Plots

To create a contour plot, select **Plot→Display...** to display the 2-D Display Control Window (Figure 2-16).

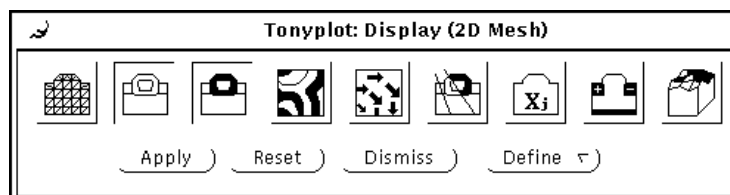


Figure 2-16: 2-D Display Control Window

We need to enable domain plotting, plot materials in different colors, plot net doping contours, and plot device junctions. To enable these features, select the icons shown in Figure 2-17.

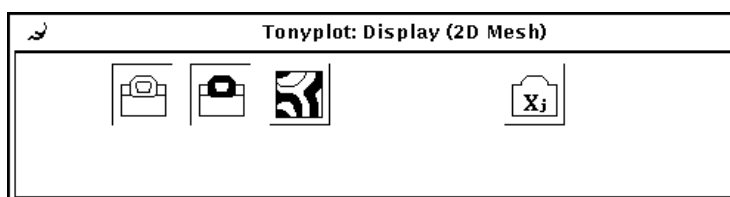


Figure 2-17: 2-D Display Icons

Next, access the Contours popup (Figure 2-18) by selecting **Define→Contours**.

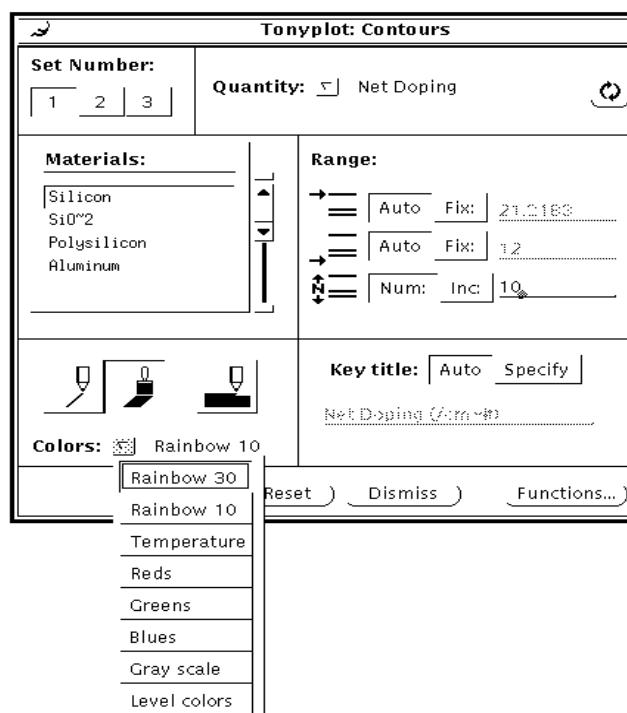


Figure 2-18: The Contours Popup

This popup shows that the contours are plotted only in silicon regions (because **Silicon** is selected on the **Materials** list), and that the contours are drawn using the color scheme **Rainbow 30**. When the definition of the contour plot is finished, select **Apply**.

As an exercise in zooming, you can examine areas of detail in this plot.

2.5.10: Interactive Cutline Plots

When a two dimensional contour plot is displayed, you can generate one dimensional profiles of any solution quantity along a cut line through the device. As an example, look at the doping profile of phosphorus along a vertical line through the LDD doping region.

Select **Tools**→**Cutline** to display the **Cutline** window as shown in Figure 2-19.

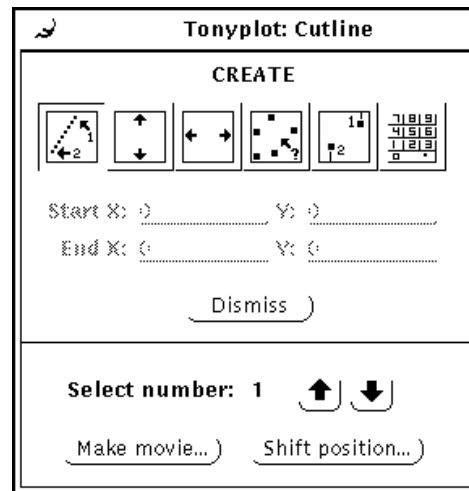


Figure 2-19: Cutline Window

Select the vertical cut option from this popup window by clicking on the second icon. Then, draw a vertical line directly onto the two dimensional contour plot with your mouse as shown in Figure 2-20.

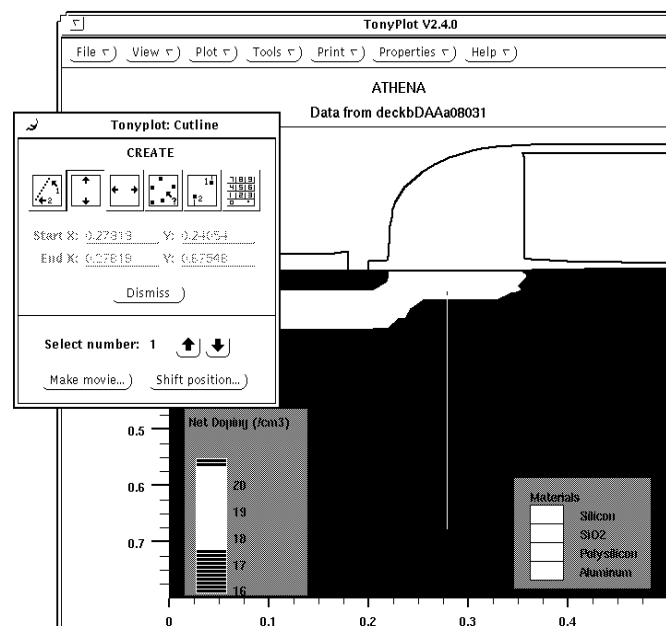


Figure 2-20: Creating Cutlines

This action causes a second plot window to appear with the one dimensional doping profile displayed.

Using the cutline control popup window again, select the **Shift position...** button and then click on the horizontal arrows. This causes the cutline to be moved around the contour plot. The cutline only moves around when the contour plot is selected.

2.5.11: Extracting Process Parameters

At this point, the process simulation of the LDD MOS transistor is essentially complete. Before continuing on to device simulation, you will use DECKBUILD's EXTRACT capability to determine some parameters related to the structure. (See also Chapter 5: "DeckBuild:Extract").

Before proceeding to the parameter extraction statements, it is a good idea to remind yourself of the present form of the simulated structure (Figure 2-21).

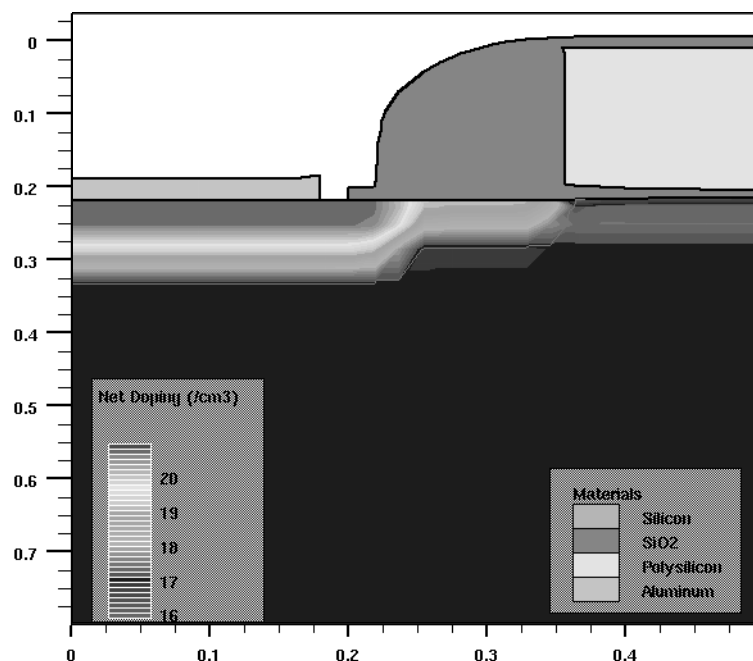


Figure 2-21: The Material Regions and Net Doping Contours of the Simulated Structure

The following parameters that are extracted in the example input deck are:

- The source/drain junction depth
- The device threshold voltage
- Conductance vs. bias
- Some sheet resistances
- The channel surface impurity concentration

Source/Drain Junction Depth

In order to extract the junction depth correctly, the extract system needs the following information:

- The name assigned to the extracted parameter. In this case, the parameter is named `nxj`.
- The name of the parameter to extract. In this case, it is the junction depth that `extract` refers to as `xj`.
- The material containing the junction. In this case, the material is Silicon. In more complicated simulations, you can create structures with stacked layers of different materials each of which may contain junctions.
- The layer occurrence number. After specifying that you are interested in the junctions in Silicon material, you have to specify which of the possibly many stacked Silicon layers you are interested in. For this structure, there is only one Silicon layer. Therefore, specifying the layer occurrence number is optional.

- Where in the source/drain region you want the junction depth to be determined. The depth of the junction varies from `xj`, the value you are looking for within the body of the source/drain region, to zero at the edge of the region. In order to extract the correct value for junction depth, a point within the body of the source/drain region must be used. In this example, the value `0.1mm` into the source/drain region is used. As shown in Figure 2-21, this value should yield the junction depth for the source drain region.
- The junction number. In complicated structures, it is possible to have more than one junction within a material layer. For example, an `n+` source/drain region within a `p` well on an `n` substrate would, on a line through the source/drain region, have two junctions. For the present structure, there is only one junction. Specifying the junction number is optional.

The statement that extracts the junction depth is as follows (this should all occur on one line):

```
extract name="nxj" xj silicon mat.occno=1 x.val=0.1 junc.occno=1
```

After this statement executes, the following value of the calculated junction depth is printed:

```
nxj=0.0987025 um from top of first Silicon layer X.val=0.1
```

This information is also written to the file `results.final` in your current working directory. When the simulation is complete, you can review the values of the extracted parameters by printing or viewing the `results.final` file.

Device Threshold Voltage

The `extract` capability is now used to calculate the threshold voltage. For this extraction, you need to specify:

- The name assigned to the extracted parameter. Use the name `nldvt` (the `n`-type, one dimensional threshold voltage).
- The name of the parameter to be extracted. For this example, threshold voltage, the parameter is `ldvt`.
- The device type. In this case, you have an `n`-type transistor, and so you specify `ntype`.
- The backbias voltage `vb`, which you set to `0V`.
- The trapped charge `Qss`, which you specify as `1e10`.
- A point along the channel where the threshold voltage is extracted. Specify the point `x=0.49`, which is just short of the right hand edge of the simulated device.

The one line `extract` statement used to calculate the threshold voltage is

```
extract name="nldvt" ldvt ntype vb=0.0 qss=1e10 x.val=0.49
```

This statement yields a result of

```
nldvt=0.671481 V X.val=0.49
```

This result is saved to the `results.final` file.

Conductance vs. Bias

The next example extracts the curve of conductance versus applied bias. This example is more complicated than the previous two examples because it involves the use of two `extract` statements. The first statement sets up the biasing conditions. The second statement extracts the conductance curve.

When multiple statement `extract` operations are performed, the `extract` system has to be told that a multi-statement `extract` operation is being defined. To do this, use the `start`, `continue`, and `done` keywords. The `start` keyword indicates the first statement of a multiline `extract` operation. The `continue` keyword indicates an intermediate line. The `done` keyword indicates the last line. For example, the statements that define a four-line `extract` operation would start as follows:


```

extract start ...
extract cont ...
extract cont ...
extract done ...

```

Use as many extract continue statements required, including (as in the example that follows) zero.

The conductance of the channel is extracted on a one dimensional line through the gate with the gate poly biased between 0 and 2 volts. The first extract statement defines the biasing conditions on the poly and is an extract start statement. Select the first occurrence of the material Polysilicon on a line through the gate ($x=0.45$), and apply the biasing conditions from 0 to 2 volts in steps of 0.2 volts:

```

extract start material="Polysilicon" mat.occno=1 \
    bias=0.0 bias.step=0.2 bias.stop=2 x.val=0.45

```

The use of the continuation character (\) at the end of the first line allows a single extract statement to span more than one line.

Once you specify the bias conditions, you can extract the conductance curve. Unlike the previous two examples where a single value was extracted (i.e., junction depth or threshold voltage), we are now extracting a curve (i.e., an array of values). The syntax for specifying a curve using extract is:

```

curve(x-axis, y-axis)

```

Here the x-axis is the applied bias, which is specified by the keyword `bias`, and the y-axis is the one dimensional n-type conductance, which is specified as `ldn.conduct`. Also specify conductance that should be used for the curve by specifying the material. In this case, it is Silicon. In a general structure, there may be more than one layer of silicon. Therefore, to make the requirement more explicit, you can state interest in the first occurrence of the material Silicon and the first region of Silicon in the device. This is done using the parameters `mat.occno=1` and `region.occno=1`. The full specification for the curve we want to extract is then:

```

curve(bias, ldn.conduct material="Silicon" mat.occno=1 region.occno=1)

```

Since there is only one silicon region in this structure, this could have been abbreviated to:

```

curve(bias, ldn.conduct material="Silicon")

```

When a single value is extracted, such as V_t , the extract system prints the value to the tty output region and logs the value to the file `results.final`. When extracting a curve, you must specify a file in which the curve is saved by specifying `outfile=value` as part of the extract statement. In this case, save the extracted curve in a file called `extract.dat`. The last statement for the conductance curve extraction is:

```

extract done name="sheet cond v bias" \
    curve(bias, ldn.conduct material="Silicon" mat.occno=1 region.occno=1) \
    outfile="extract.dat"

```

After performing the extraction, you can plot the conductance curve shown in Figure 2-22 by selecting the file named `outfile`, invoking TONYPLOT through the **Tools** menu of DECKBUILD.

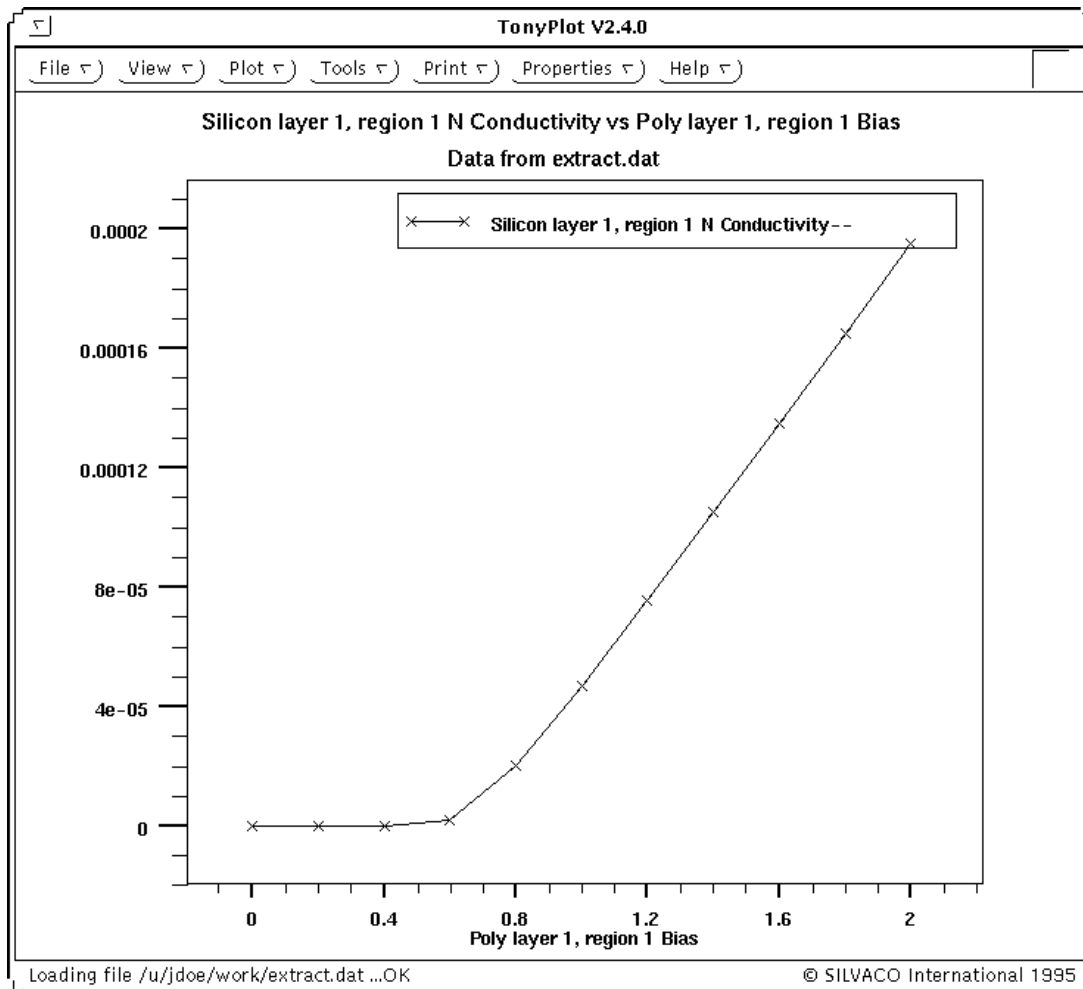


Figure 2-22: The Conductance Curve

Sheet Resistances

The n++ Source/Drain Sheet Resistance

The statement that causes extraction of the n++ source/drain resistance is similar to the extraction statement used for the junction depth. It supplies the following information:

- A name for the extracted parameter. In this case, the parameter is named `n++ sheet rho`.
- The name of the parameter to extract. For the junction depth you specified `xj`. For the sheet resistance, specify `sheet.res`.
- The name of the material containing the n++ region. In this case, it is silicon.

To be sure you have the correct material layer, specify the material occurrence number and the region occurrence number. Since there is only one silicon layer in the structure, it is unnecessary to give this information here. This information, however, is required for more general structures.

- Finally, you must tell the extract system where the n++ region is by giving the location of a point within the region (take `x=0.05`).

The extract statement for the sheet resistance is then:

```
extract name="n++ sheet rho" sheet.res material="Silicon" \
mat.occno=1 x.val=0.05 region.occno=1
```

When `extract` executes this statement, it will print the following value for the calculated sheet resistance:

```
n++ sheet rho=39.9388 ohm/square X.val=0.05
```

This information is also written to the file `results.final` in your current working directory.

LDD Sheet Resistance Beneath The Oxide Spacer

To extract the sheet resistance under the oxide spacer, simply move the point of interest to under the spacer. For the present structure, a value of 0.3 is reasonable. Name the extracted resistance `ldd sheet rho`:

```
extract name="ldd sheet rho" sheet.res material="Silicon" \
mat.occno=1 x.val=0.3 region.occno=1
```

The extracted value is printed after execution:

```
ldd sheet rho=2771.32 ohm/square X.val=0.3
```

Channel Surface Impurity Concentration

The final parameter that is extracted before leaving the process simulation section is the net doping concentration at the surface in the channel. In the case of junction depth, you specify `xj` as the extraction target, while for the sheet resistance, you specify `sheet.res`. For surface concentration, specify `surf.conc` as the extraction target, and specify the name of the impurity: `impurity="Net Doping"`. A point within the channel is given using `x.val=0.45`. The full extraction statement for the net doping channel surface concentration is then:

```
extract name="chan surf conc" surf.conc impurity="Net Doping" \
material="Silicon" mat.occno=1 x.val=0.45
```

The extracted channel surface concentration is printed after execution:

```
chan surf conc=2.78719e+16 atoms/cm3 X.val=0.45
```

which is also logged to the file `results.final`.

2.6: Completing The Device Structure

Although the process simulation of the LDD MOS transistor is complete, there is some additional work in ATHENA to be done before moving on to device simulation. When symmetry is exploited during process simulation, the expanded full structure must be assembled within the process simulator. Once the full device is available, the device electrodes can also be specified within ATHENA.

The calculations performed so far in this tutorial have calculated the shape of a half-device. The full device is generated by reflecting the calculated structure about the axis of symmetry through the center of the gate.

ATHENA structure statements are used to assemble a complete structure. In this example, the structure must be reflected about the right-hand boundary. This is achieved by the statement:

```
structure mirror right
```

Source, gate, drain and substrate electrodes for the full device are now specified using an electrode statement for each of the polysilicon and aluminum contact materials:

```
electrode name=gate x=0.5 y=0.1
electrode name=source x=0.1
electrode name=drain x=0.9
electrode name=substrate backside
```

The device structure is now ready for simulation by ATLAS. It is good practice to save the structure at major milestones during the simulation. The completion of the process simulation is a major milestone, and the example input deck saves the structure to the file `mos1ex01_0.str`.

If you are performing simulation in parallel with reading through the tutorial, you should now continue the simulation to the point where the structure is plotted by TONYPLOT. The plot displayed by TONYPLOT should look like the plot shown in Figure 2-23.

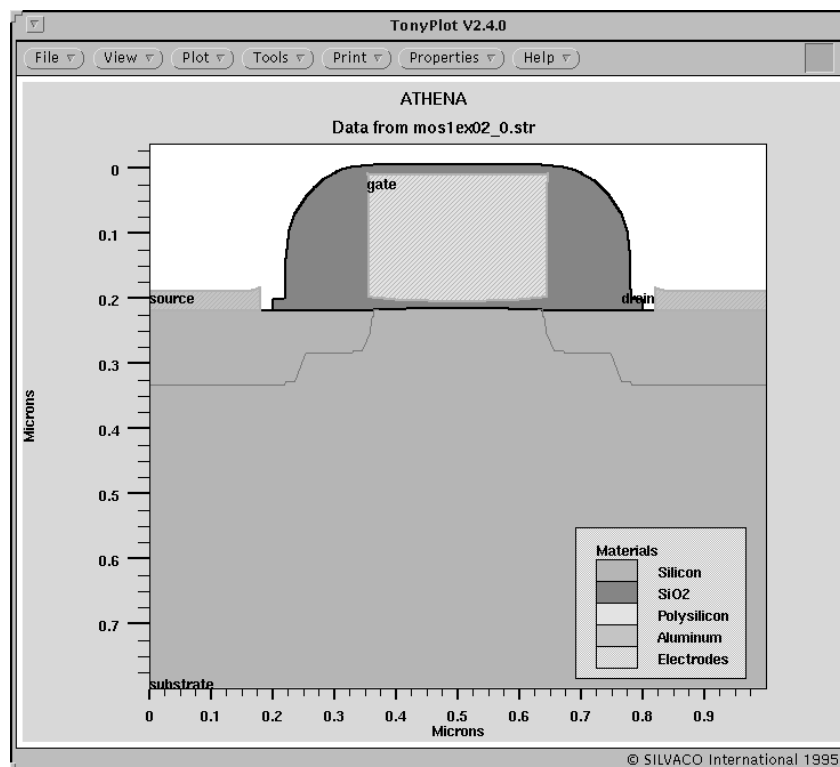


Figure 2-23: Process Simulation of the Transistor Structure

2.7: Device Simulation

Now calculate an Id-Vgs characteristic for the structure with Vds held constant at 0.1V. After performing the calculation, extract the parameters Vt, Beta, and Theta, which are often used to characterize the behavior of MOS transistors. The auto-interfacing statement, `go atlas`, causes the execution of ATHENA to stop and the execution of ATLAS to start, using the structure defined most recently in ATHENA.

2.7.1: Defining The Device Physics

The next task is to define the physical models that ATLAS includes in its calculations. The strategic choices include:

- whether to perform a bipolar or a unipolar simulation,
- what transport model to use (i.e., drift diffusion or energy balance),
- what generation and recombination processes (if any) to account for.

After making these strategic decisions, the remaining choices center around the specification of model parameters.

The general case is to include both electrons and holes. This choice is required if any generation or recombination processes are to be accounted for. If no generation and recombination processes need to be accounted for, it may be acceptable to account for only one carrier which in result in faster calculations. In this example, both electrons and holes are included by specifying `numcarr=2`.

Model carrier transport in the drift diffusion approximation, which is the default. Other options include solving for electron and hole carrier temperatures or solving for lattice heat flow or both. Use a general purpose mobility model that includes the effects of carrier concentration, temperature, and dependencies on both the parallel and transverse electric fields. dependencies. This model is named `cvt` in ATLAS. Also, include the effect of Shockley-Read-Hall (SRH) recombination using the ATLAS `srh` recombination model.

Define the gate contact to be n-type polysilicon, and define the work function associated with the gate using the command

```
contact name=gate n.poly
```

Define the fixed charge at the Silicon/Oxide interface to be 3×10^{10} using the statement `interface qf=3e10`.

Printing of status information to the tty region of DECKBUILD is enabled using the pseudo-model print.

These models and options are selected using are selected using the command

```
models cvt srh print numcarr=2
```

2.7.2: Performing The Device Simulation

The LDD MOSFET is now ready for device simulation, and its electrical characteristics can be calculated for specified bias conditions in ATLAS.

In this example, calculate an Id-Vgs curves, for Vgs between 0 V and 3.0 V, with Vds held constant at 0.1 V. From this data extract, several parameters that are used to characterize the behavior of MOS transistors.

The device simulation proceeds in three steps. The first step is to obtain an initial solution at zero bias. The second step is to apply the constant drain voltage with zero gate bias. The third step is to increment the gate voltage with the drain voltage held constant.

The first step is accomplished with the `solve init` statement, which solves the device at zero bias.

```
solve init
```

In the second step, solve to reach the final drain voltage of 0.1V by stating the drain voltage. Voltage increments greater than 0.1 V are not recommended for initial voltage steps.

```
solve vdrain=0.1
```

The Id-Vgs curve can now be calculated. To retain this data for additional processing arrange, collect it in a log file by using the following log statement:

```
log outf=moslex01_1.log master
```

This saves the simulated curve to the file moslex01_1.log.

The statement that causes the Id-Vgs characteristic to be calculated is:

```
solve vgate=0 vstep=0.25 vfinal=3.0 name=gate
```

Once you perform the simulation, then plot this file using TONYPLOT. The graph displayed by TONYPLOT should look like Figure 2-24.

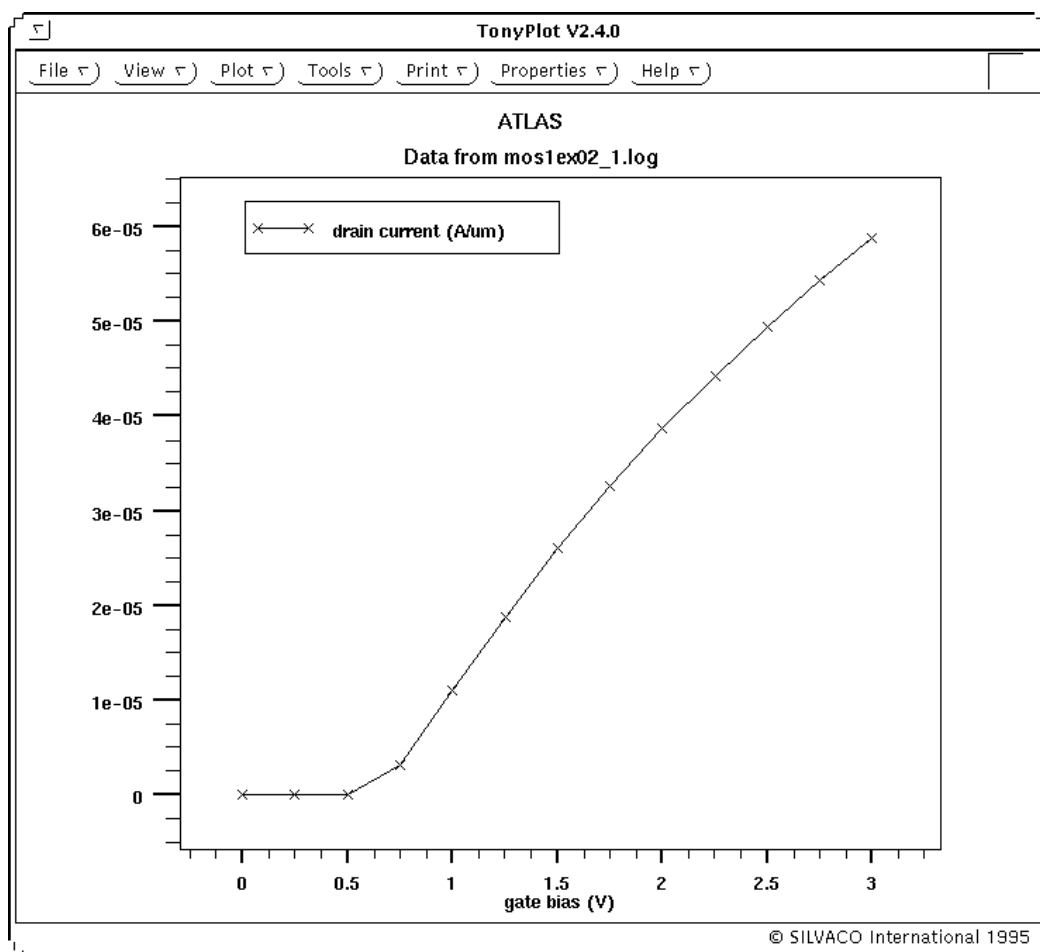


Figure 2-24: Device Simulation of the Structure

For more information about ATLAS, see the ATLAS Tutorial in the ATLAS USER'S MANUAL.

2.7.3: Extracting Device Parameters

Several device parameters that are used to characterize the electrical behavior of MOS transistors are now extracted. These parameters are V_t , Beta, and Theta. The `extract` statements associated with these parameters may appear intimidating but this does not matter. The statements were automatically generated from `extract`, which was selected from the **Commands** menu of DECKBUILD while ATLAS was running.

The execution of the `extract` statements should yield the following values (but may differ slightly):

```
nvt=0.529356
```

```
nbeta=0.000243389
```

```
ntheta=0.13192
```

2.8: Going Further

Now you completed this tutorial, the next step should be to start using DECKBUILD, TONYPLOT, ATHENA, and ATLAS to perform simulations. You can gain more experience by loading, running, modifying, and experimenting with a selection of the examples that are provided with DECKBUILD.

Depending on your applications, you may never need to go beyond this basic level of use. Many, however, wish to proceed (sooner or later) to more advanced use of semiconductor technology CAD. As you go further, you will find there are at least two more major milestones ahead. Refer to these as intermediate and advanced levels of use.

Intermediate level users exploit the full range of features provided by DECKBUILD and TONYPLOT and by the other VWF INTERACTIVE TOOLS described in the DEVEDIT, MASKVIEWS, and OPTIMIZER chapters. In addition, you must acquire broad familiarity with all of the features, options and capabilities of ATHENA, ATLAS, MERCURY, and CLEVER or whichever simulator is used. See the respective user's manuals for more information. There are tutorial chapters in each of these manuals.

Advanced users exploit all the levels of the VIRTUAL WAFER FAB (VWF) environment. The VWF INTERACTIVE TOOLS are the first level of this environment. The second level (VWF AUTOMATION TOOLS) makes it convenient to design, define, and run large simulation-based studies in a way that uses networked and parallel computing environments. The third level (VWF PRODUCTION TOOLS) makes it easy to analyze simulation data in various ways that provide guidance and insight to engineers involved in production and manufacturing.

3.1: Starting Manager

The File and Applications Manager (MANAGER) is a utility that allows easy access and interfacing to the INTERACTIVE TOOLS and their associated files. It displays all the tools available and all the relevant files in the current UNIX working directory and allows tools to be started with selected files automatically, providing the correct parameters are used.

You can start MANGER from the UNIX command prompt by entering:

```
manager
```

The MANAGER window will then appear (Figure 3-1). This assumes that the `$SILVACO` environment variable has been defined to point to the directory into which the SILVACO software was installed and that `$SILVACO/bin` is included in the user's search path. There are no command line options for the MANAGER. If this is the first invocation of MANAGER, the following message appears:

```
Creating new configuration files.  
This will take a few moments, please wait...  
Configuration files created.  
Starting Master Manager
```

A `.silvaco` directory is then created in the user's home directory, and a copy of the MANAGER configuration files is placed in the `.silvaco` directory. This directory is used for subsequent invocations and allows each user to customize their setup.

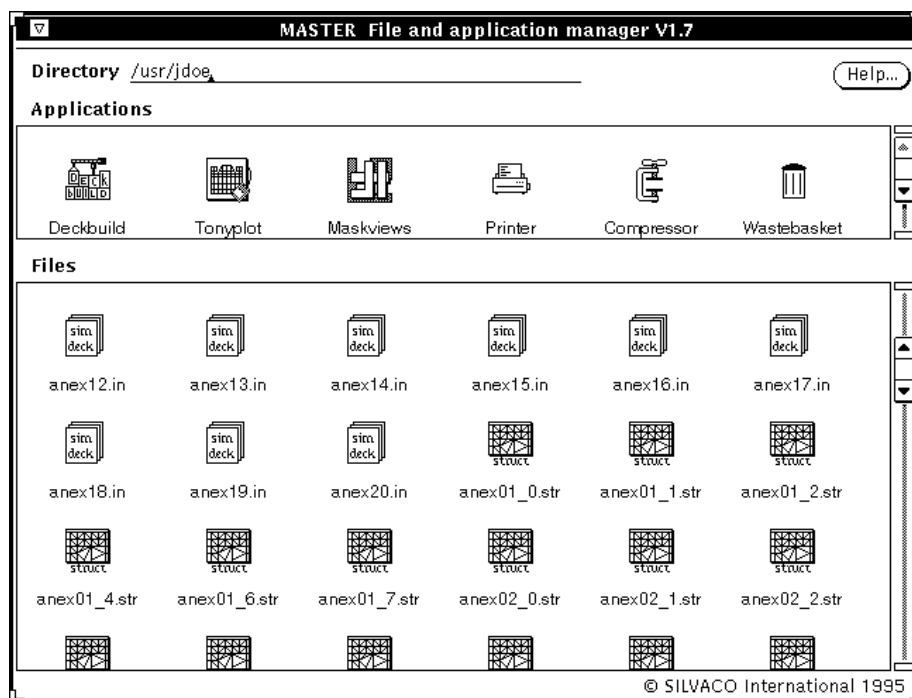


Figure 3-1: The Manager Window

3.2: Using Manager

The user interface of **MANAGER** allows easy use and access to the **INTERACTIVE TOOLS** and files. Once started, a window is displayed showing the current directory name and two large windows containing several icon images representing the available applications and files.

3.2.1: File and Application Windows

The upper window on the main display is the applications window. All of the currently defined **INTERACTIVE TOOLS** applications are listed as a set of icons with the application name written beneath the icon. Other useful applications and utilities may also be listed here. Some of the applications normally listed here are **TONYPLOT**, **DECKBUILD**, **MASKVIEWS**, **DEVEDIT**, and **SPDB**.

The lower (normally larger) window contains a list of all of files in the current working directory which are recognized by **MANAGER**. This always includes other directory names, including the parent (**..**), and most of the files known to be used with any of the **INTERACTIVE TOOLS** simulators, tools or utilities. This window updates automatically whenever files are created or deleted from the displayed directory.

As the mouse pointer moves over icons on the file window, the type and name of the file is shown at the foot of the main window.

3.2.2: Changing the Current Directory

You can change the current working directory by using either of the following methods.

1. Type a new directory name in the **Directory** field at the top of the **MANAGER** window and press the Return key. The file window will be updated with all of the recognized files contained in the new directory.
2. Double-click on one of the directory icons. The files window is then updated with the new directory's contents and the path name displayed at the top of the window is updated with the new directory name. The **..** directory is used to move up a level to the current directory's parent in the file system hierarchy.

3.2.3: Starting Applications

To start applications, either double-click on an application icon or drag and drop one or more files onto an application. For example, if you want to load **TONYPLOT** and display the structure file **mos1.str**, drag the **mos1.str** icon onto the **TONYPLOT** icon. If you want to load **TONYPLOT** and display the structure files **mos1.str** and **mos2.str**, select the structure files, and drag them onto the **TONYPLOT** icon.

MANAGER always uses the correct command line switches for the specified files.

3.2.4: On Line Help

The on-line manual (displayed by pressing the **Help...** button on the top of the main window) provides an in-depth description of the function and operation of all **MANAGER** features. When first displayed, the on-line manual shows an index of all the major sections available. The buttons along the top of the Help Display are used to navigate between the pages of the manual. Selecting Return to index causes the initial main section index to be displayed. The **Section** button contains a menu of all the major sections in the manual. If you one of these sections, then the topics for that section will be displayed. You can then use the **Sub-section** button to select a manual page to be viewed. To print out the manual pages, click on the **Print** button on the help display.

3.2.5: Customizing

To use the customization options, select the **Attributes** option from the menu displayed by clicking on the MENU mouse button while the pointer is over the files display window. The **Attributes** popup will appear (Figure 3-2). This allows you to change the attributes of the displayed files.

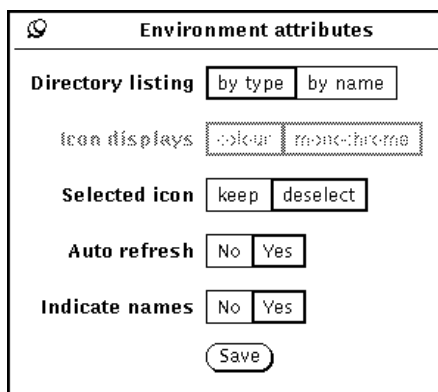


Figure 3-2: Attributes Popup

The following attributes can be customized:

- **Directory listing** — Specifies whether to sort the icons in the file window in order of their file type (by type) or alphabetically (by name).
- **Icon displays** — Allows the selection of color or monochrome icons when used on color display systems. This option is not available on monochrome systems.
- **Selected Icon** — If you select **keep**, then file icons highlighted for use with applications will remain highlighted after the application has been started. If you select **deselect**, then the file icon does not remain highlighted.
- **Auto refresh** — This toggles the **Auto Refresh** function. **Auto Refresh** periodically scans the current directory, checks for files that were created or removed, and updates the files window accordingly.
- **Indicate names** — Specifies whether the name and type of the file will be displayed at the foot of the window as the pointer passes over the icon(s).

Clicking on the **Save** button saves all of the options in the file `$HOME/.masterrc`. The saved attributes will be used the next time MANAGER is started.

3.2.6: Applications

The **Applications** popup (Figure 3-3) is used to configure the Silvaco applications.

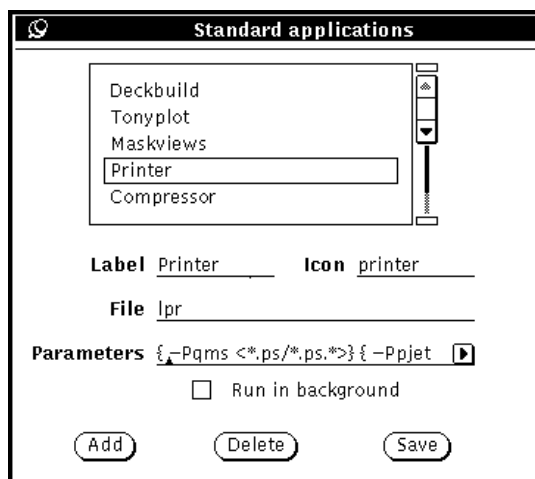


Figure 3-3: Applications Popup

To access the **Applications** popup, click the right mouse button and to cause the **Options** menu to appear. Then, select the **Control** submenu and select **Applications**.

The list box shows all of the available application names. To configure an application, select the desired application from the list box, change one or more of the following fields, and select **Save**. Here are the following fields.

- **Label** — This field specifies the name that is displayed on the Applications Window.
- **Icon** — This field specifies the name of the icon used to generate the displayed image. Two types of icon files are needed. The `.icon` file is used to draw the black outline image of the icon, and the `.fill` file is used to draw the colored fill regions of the icon.
- **File** — This field specifies the command to be executed when the application is selected. Enter the full path name of the command if the application is not likely to be in the user's normal search/path. For example, the `$PATH` environment variable.
- **Parameters** — This string is used to select which parameter switches are used with specific file types for each application. Each parameter switch is listed inside a pair of braces (`{}` and `}`) with a list of the file types to be used with that switch. File types to be used with the switch are contained within angle brackets (`<` and `>`) and are specified in terms of UNIX search strings with `?` and `*` used as wild cards. Files for which the switch must not be used are listed between a set of brackets (`[` and `]`). File types between brackets are separated by a forward slash (`/`). For example, if you want to specify a command switch option that would be used with all files, which ended in `.std` or `.str` and did not contain a `Z` (i.e., the file `mosz.str` would be omitted), then enter the line `{-st <*str/*std>[*Z*]}`.
- **Run in Background** — This option is used to select whether **MANAGER** waits for the application to complete before continuing. If **MANAGER** runs the application in the background, then it continues without waiting for the application to finish.

The **Color** choice field (on color systems only) allows the color used in the fill area of the icon image to be selected. The **Add** button copies the currently selected item to the end of the list so you can add or modify applications. The **Delete** button removes an item from the list.

Clicking on the **Save** button causes the applications setup to be saved. The saved file is stored in the `$(HOME)/.silvaco` directory.

Note: The configuration file formats have changed slightly from Version 1.5 to Version 1.6 of Manager. Using the old format files with the new version of the program may cause spurious colors for the icons to be displayed.

3.2.7: File types

You can edit the file types listed in the file window using the File Type popup (Figure 3-4). To open this popup, select **File types** from the **Control** submenu. The scrolling list contains entries for all of the file types that are displayed. You can edit an item by selecting one of the rows in this list. Other items are as follows:

- **File type** — This field is a text string that is displayed at the foot of the window as the pointer passes over files of this type in the Files Window.
- **File mask** — This field is used to mask which files from the current directory are to be displayed as this particular type. UNIX search strings separated by the forward slash (/) character are entered here for all file types. For example, if a file type contains all files ending in `std` or `str`, then the file mask string would be `*std/*str`.
- **Default** — This field is used to automatically start an application when double-clicking on an icon of this type. The string entered here should be an application name and command line switches so that the correct command line is created once the file name is appended to the end of the string.
- **Icon, Color, Add, Delete, and Save** — These fields are the same as those described in the Applications Window (see Section 3.2.5: “Customizing”).

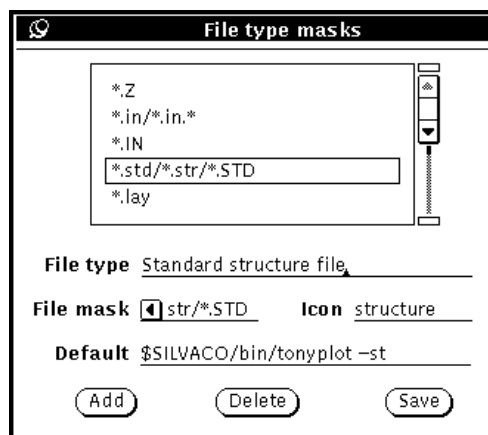


Figure 3-4: File Type Popup

This page is intentionally left blank.

4.1: Introduction

4.1.1: Overview

DECKBUILD is an interactive, graphic runtime environment for developing process and device simulation input decks. It consists of a window for input deck creation and editing, a window for simulator output and control, and a set of popups for each simulator that provide full language and run-time support.

4.1.2: DeckBuild's Purpose

DECKBUILD is an extremely powerful and flexible tool that is easy to use and provides many automated features that previously required user operation. Among these features is DECKBUILD's ability to generate error-free simulator syntax driven from user-friendly popup windows. This feature allows for transparent transition from one simulator to another, automatic definition of mesh and mask information, and application of built-in measurement (extraction) facilities. Before DECKBUILD, these tasks often required user intervention and were extremely time consuming. By automating these tasks, DECKBUILD allows you to concentrate on the real work at hand: accurate simulation.

4.1.3: Features

DECKBUILD also offers several powerful facilities never before available. One of these facilities, the global optimizer, allows optimization across an entire input deck even between different simulators. For example, varying an implant dose in SSUPREM3 and a diffusion time in ATHENA permits optimizing against a Vt curve simulated with ATLAS. DECKBUILD also provides a seamless integration with DEVEDIT and its adaptive meshing capabilities. In addition, the UTMOST interface allows Silvaco's parameter extraction package UTMOST III to load data from one of more device simulation runs and perform SPICE model parameter extraction. DECKBUILD offers real flexibility with the ability to use UNIX system commands within simulation decks and the added feature of executing simulations on remote hosts while DECKBUILD is running locally.

Two new features are the communication interfaces to the new MERCURY interface tool and EXACT. When a MERCURY tool is started from DECKBUILD, a communications pipe is enabled to allow MERCURY decks to be written directly into DECKBUILD for execution. EXACT uses the new input/output pipe command-line options to send a simulation deck for execution. EXACT then receive extracted results from DECKBUILD.

Note: Pipe in this context means a conduit that transfers data between two program.

DECKBUILD also contains many other convenience features:

- A built-in tool palette allows interactive plotting of the current structure.
- Instant substitution of a different cut-line set from the layout editor.
- Full interactive control of the simulator, including a history function that allows you to back up in the deck and try again.
- Interactive cut-and-paste to either the simulator or the text editor.
- A display in the input deck of the currently executing line and other features.

Simulators

Many simulators are available in the DECKBUILD Library and most are supported by a complete set of interactive popup windows. By selecting or moving various items on each popup, you can easily generate correct syntax. On-line context-sensitive help is also available. A deck is built by going through each desired popup and clicking on a **WRITE** button. This causes syntax to appear in the text editor. The deck can be saved and retrieved for later use. The popups have the additional feature of input-deck parsing. To do this, highlight a section of the input deck and choosing **Parse Deck**. All appropriate popups will then re-configure themselves to reflect the syntax. For example, if you highlight an ATHENA IMPLANT statement and press the **Parse Deck**, the ATHENA Implant popup will appear. This popup would reflect the values in the highlighted syntax.

For manual deck editing, DECKBUILD has a built-in text editor. The text editor allows easy point-and-click editing, cut and paste to and from any other window, find/replace, multiple scroll views, and other features.

Autointerface

DECKBUILD allows and encourages concatenating of decks from different simulators. For example, a simulation can start with SSUPREM3 for fast 1-D process simulation, move into ATHENA for 2-D process simulation, and be followed by any number of separate ATLAS device tests. Figure 4-1 shows a schematic of this flow. The entire result is saved as a single input deck.

Notice how process simulation is treated as a serial flow of events, while device simulations are treated as parallel. This is because of the way in which auto interfacing works. At the conclusion of each process run, the simulation results are saved and are used by the next process simulator; several process decks form a serially-linked chain. Device tests always use the last available process result. Auto interfacing is one of the most powerful features in DECKBUILD.

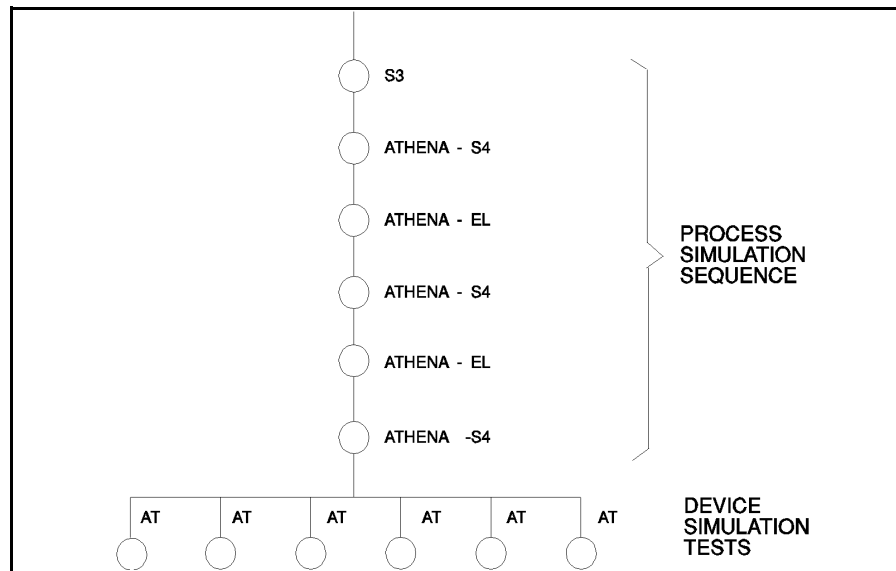


Figure 4-1: DeckBuild Schematic

Execution Control

DECKBUILD provides a diverse set of controls over the running simulation. You can run the entire deck. You can run it one line at a time. You can run the deck until a predefined line is reached or halted immediately after the current command. You can even set multiple break points in the deck by using the **String Monitor** option and by adding a specific comment to search for at the required locations. While the simulation is running, DECKBUILD also highlights the currently executing line in the input deck. The simulator itself can be stopped, started, quit, killed, paused, and unpaused.

One of DECKBUILD's unique features is the **History** function (see Section 4.9: "History"). DECKBUILD remembers each line of the deck as it is executed and saves a structure file after each one. As a result, if a problem is discovered it is unnecessary to re-do the entire deck from the start. For example, if after running part way down an input deck and you discover a missing statement or an erroneous value, you only need to point and click on the line from which to start and click on **Init from History**. DECKBUILD automatically re-loads the saved history file and allows the simulation to continue from that point.

DECKBUILD also allows plotting the current structure. At any point in the deck, click a button and DECKBUILD automatically causes the simulator to save a structure file, then start up SILVACO's post-processing tool using the saved structure as input. This is often useful in conjunction with **History** to aid in fast tuning of a section of input deck. A statement can also be changed then re-executed, and the change is immediately visualized.

Examples and Tutorial

DECKBUILD provides full on-line examples that can be loaded up at the press of a button. These examples provide input decks for actual devices and help when learning about DECKBUILD. Chapter 2: "Tutorial" is a tutorial that explains how to use DECKBUILD to perform a simple simulation.

Advanced Topics

Generic Decks

Most decks have built-in geometric constants that reproduce a single, unchangeable cross-section of a wafer. DECKBUILD's IC layout interface (MASKVIEWS) makes it possible to write a single deck that can be used at any location on a wafer without using hard-coded geometry information. You can create (or read from GDSII or CIF format) device layout and mask layers using MASKVIEWS. Then, create or modify a deck using DECKBUILD to use mask names rather than deposit and etch statements with hard-coded geometry values. Finally after making a cutline using MASKVIEWS, DECKBUILD can simulate that cross-section. You can simulate any cross-section of the device in this manner.

Extraction

DECKBUILD contains built-in extract routines for both process and device parameter extraction. EXTRACT forms a "function calculator" that allows you to combine and manipulate values or entire curves quickly and easily. You can take one of the standard expressions and modify it as appropriate to suit your needs or use the custom extract language to create unique extraction statements specific to the current simulation.

EXTRACT also includes features, such as variable substitution and internal 1D device simulators, QUICKMOS and QUICKBIP for specialized cases of MOS and bipolar electrical measurement. All extracted results are displayed in the DECKBUILD TTY subwindow and stored in a datafile for easy comparisons of different simulations (See Chapter 5: "DeckBuild:Extract").

Optimization

A powerful OPTIMIZER is available within DECKBUILD that allows quick and accurate tuning of simulation parameters. Specify any number of input parameters to vary and any number of targets to attain. For example, it is possible to find a target threshold voltage of 0.75 volts by varying gate oxidation time and V_t adjust implant dose.

Optimization parameters may come from any simulator in the simulator library, and targets from any extracted parameter. For example, it is easy to set up a deck that auto interfaces from SSUPREM3 to

ATHENA to ATLAS. Then, extracted values can be optimized from I-V curves while using SSUPREM3 or ATHENA diffusion coefficients or both as input parameters. Simple graphical worksheet control with interactive runtime display of the optimization in progress makes the optimizer easy to use. In addition, the OPTIMIZER requires no modification of any kind to the input deck.

4.2: QuickStart

4.2.1: Overview

Although DECKBUILD may seem complicated at first, a deck can be built and run using only a few basic controls. If new to DECKBUILD, please follow this tutorial guide. The remainder of the manual can be read for more details on using DECKBUILD after getting familiar with its operation.

4.2.2: Starting DeckBuild

This section explains how to start DECKBUILD, create an input deck, run the deck, do an interactive plot of the results, and save the input deck file to disk. In the following startup instructions, if not running C-Shell, do not use the ampersand (&) at the end of the entered command line. The & tells csh to run DECKBUILD in the background.

Since a SSUPREM3 input deck is built in this tutorial, start DECKBUILD with the default simulator set to SSUPREM3 by entering the command:

```
deckbuild -s3 &
```

The simulator can also be selected or changed after DECKBUILD has been started from the Main Control popup.

In a few moments, DECKBUILD will appear on the screen. If it doesn't or if any error messages appear, refer to the SILVACO INSTALLATION GUIDE to verify that DECKBUILD was installed correctly.

4.2.3: Writing a SSUPREM3 Input Deck

Click and hold the **MENU** mouse button over the **Commands** menu button to display the SSUPREM3 Commands menu. It will appear as shown in Figure 4-2, showing all the buttons at the top of the frame.

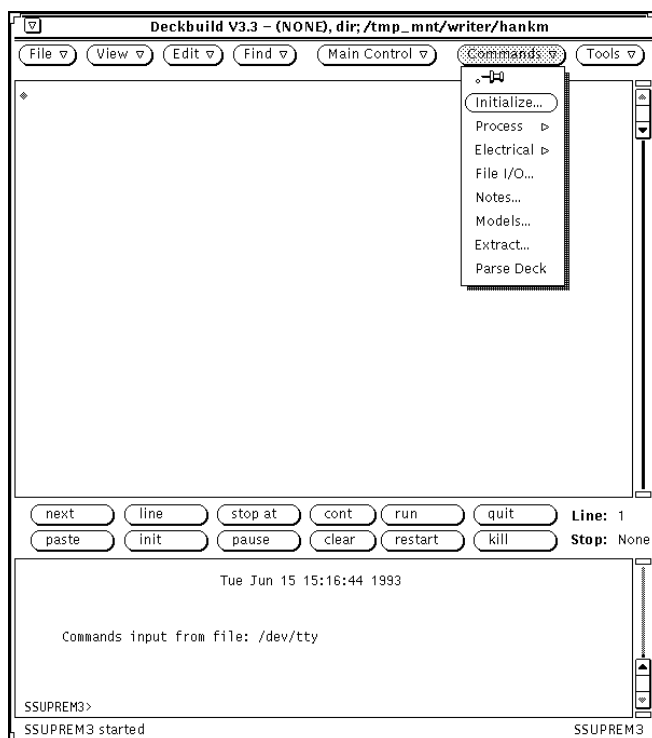


Figure 4-2: Deckbuild SSUPREM3 Commands Menu

Next, move the pointer over the **Initialize...** menu item and release **MENU**. The SSUPREM3 Initialize popup will appear as shown in Figure 4-3.

Figure 4-3: SSUPREM3 Initialize Popup

Choose the material, orientation, thickness, and grid layers by using the pointer and mouse buttons. The default values of these controls have been chosen so that, in many cases, it is unnecessary to change the value of every control on the popup. Just change the values needed. Use lists of items such as **Material** by clicking **MENU** while the pointer is over the square box. A list of materials, resembling a menu, appears. Move the pointer over the desired material and then release **MENU** to activate it.

Change the value of **Thickness** by clicking and holding **SELECT** with the pointer in the small grey slider box, then dragging the box left or right. Release **SELECT** when done. Notice that as the thickness is changed, the grid layers slider automatically follows along to maintain a constant thickness per grid layer ratio. If you not satisfied with the number of layers, change it after the thickness is set. Choose the **Orientation** by clicking **SELECT** over the desired orientation value.

The box containing the value will be indented.

To specify an initial impurity, click **SELECT** in the checkbox to the left of the impurity. The impurity name and slider become active. When the checkboxes are not checked, the impurity information becomes inactive and appears grayed out. Slide the concentration slider to the desired value. Choose an exponent for the concentration by clicking **MENU** over **Exp**. A list of exponents appears. Release **MENU** over the desired value. Finally, write the SSUPREM3 INITIALIZE statement to the deck by clicking on the **WRITE** button.

A line similar to this appears as:

```
INIT SILICON ORIENT=100 THICK=4.00 SPACES=800
```

In this fashion, you can build a process sequence by serially invoking popups from the Commands menu. Most of the commands needed are on the Process pullright menu (Figure 4-4). For example, to open the SSUPREM3 Diffuse popup, click on and hold **MENU** on the **Commands** menu button. The Commands menu appears. While still holding **MENU**, move the pointer down to **Process**.

Note: There is a small, right-pointing triangle to denote a pullright menu. Slide the pointer over to the right a few millimeters, and the Process menu will appear. Move the pointer down to **Diffuse...** and release **MENU** to invoke the popup.

You can pin the Commands, Process, and other menus for convenience. A pinned menu stays in place on the workspace and does not disappear after choosing an item, unlike an unpinned menu. To pin a menu, first invoke it with **MENU**. Then while still holding **MENU**, move the pointer over the pushpin shown at the upper left corner. The pushpin slides into its hole as the pointer moves to the left.

Invoke items from a pinned menu by clicking **SELECT** over the desired item. To unpin a menu, click **SELECT** over the pin. The pin is removed and the menu disappears.

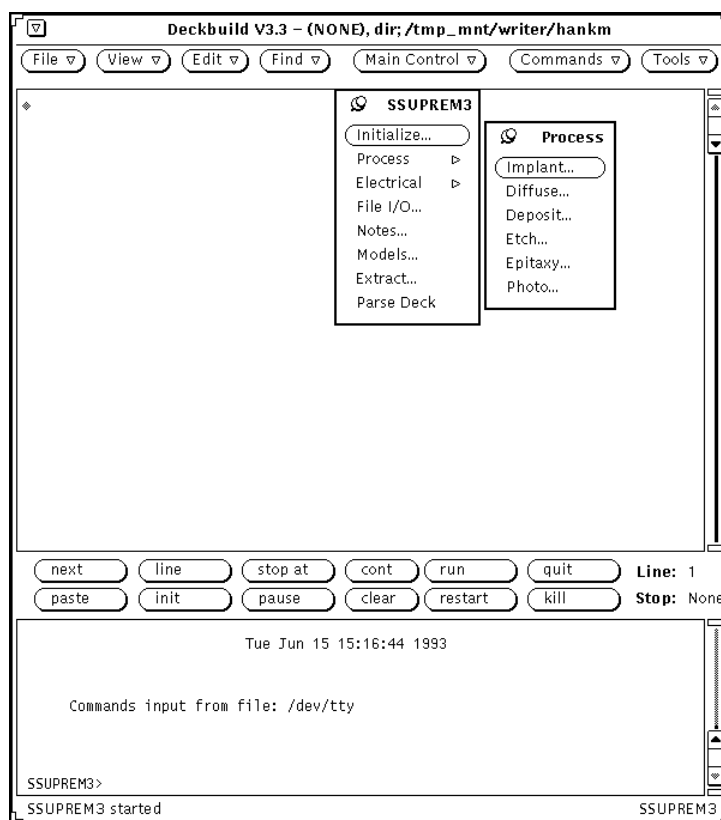


Figure 4-4: Pinned Commands and Process Menus for SSUPREM3

4.2.4: Running A Deck

A SSUPREM3 deck fragment for initial MOS processing is shown in Figure 4-5, which shows this deck in DECKBUILD. Because DECKBUILD was started with the `-s3` option, SSUPREM3 should be running and displaying a prompt in the tty subwindow. If so, you can now run the deck.

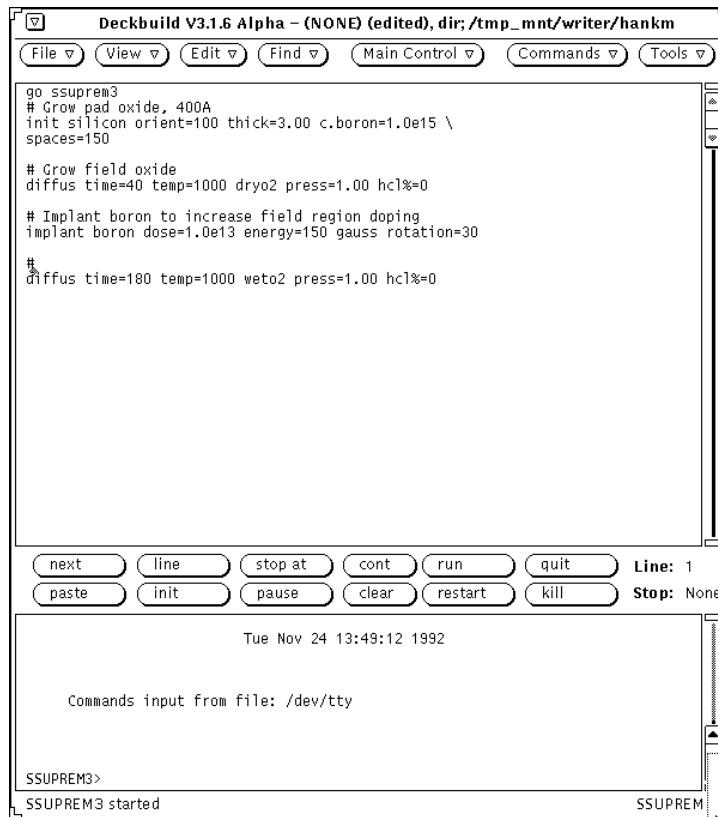


Figure 4-5: SSUPREM3 Deck Fragment

DECKBUILD permits several ways to run the deck: one line at a time, all of the input deck, or only up to a breakpoint. Run the deck a line at a time by clicking **next** on the execution control panel.

The first line (`go ssuprem3`) is ignored by the simulator. This line is an autointerface statement that tells DECKBUILD to run the following deck in SSUPREM3. Click **next** again. The first comment line is executed by SSUPREM3. SSUPREM3 just repeats the comments it sees. Click **next** again to execute the `INIT` statement. Continue in this fashion to go as far as desired into the deck.

At any time, you can tell DECKBUILD to run from the current line to the bottom by clicking **cont**. When **run** is clicked, DECKBUILD always runs from the first line to the last. Be careful if in the middle of the deck, since DECKBUILD starts running from the top again. Use **cont** to keep going from the middle of the deck.

DECKBUILD saves special files automatically after each process step. These are history files, which allow backing up and re-starting the simulation from any point in the deck. See Section 4.9: "History" for information about how it works.

Resetting The Current Line

Notice that as the deck is stepped through, the **Line** field is incremented. **Line** shows the current line in the input deck. Since **next** always operates on the current line, it is necessary to reset it when you reach the bottom of the deck if you want to starting from the top again. In fact, the current line can be set to any line in the deck, not just the top line. Set the current line by highlighting part or all of the

line in the input deck and click **Line**. The **Line** field will be updated to show the new current line number.

Plotting The Current Structure

DECKBUILD can plot the current device at any point in the process while stepping through the deck. This permits you to visualize what the device looks like at any point. For example, after a gate oxide step in a MOS device, you may want to investigate the oxide thickness, or look at a doping profile after an implant step.

To plot the current structure, first un-highlight any highlighted text on the screen. An easy way to do this is to single-click **SELECT** anywhere in the input deck. Then, click **SELECT** on the **Tools** menu button. DECKBUILD saves the current structure from the simulator then starts TONYPLOT on the structure. In a few moments, TONYPLOT will appear.

Note: Single-clicking **SELECT** on a menu button automatically chooses the default item on the menu, which is **Plot structure**.

TONYPLOT can be kept on the screen for as long as necessary. You can run additional TONYPLOTS later in the process flow to show how the formation of the device progresses. You can also show layout files using the MASKVIEWS layout editor. To start MASKVIEWS, select it from the **Tools** menu.

Saving The Deck

When satisfied with the results of the simulation, save the input deck by choosing **Save as...** from the **File** menu. The **Save As** popup will appear (Figure 4-6).

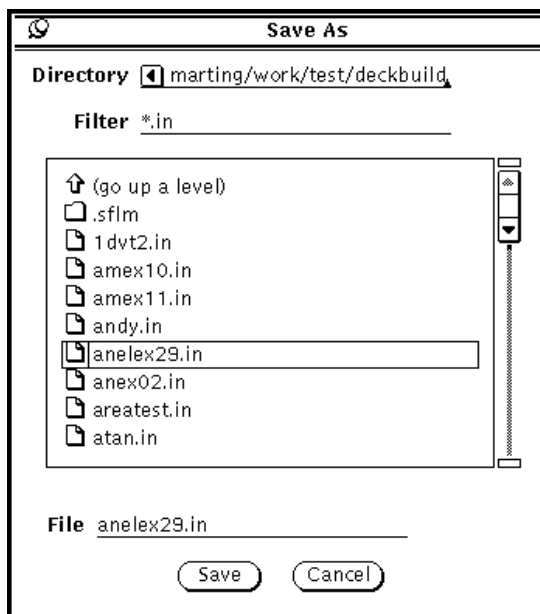


Figure 4-6: Save As Popup

Enter in the directory and the file name wanted and click **Save**. After DECKBUILD saves the file, the popup will disappear.

4.2.5: Quitting DeckBuild

The easiest way to quit DECKBUILD is to select **File→Quit**. This displays a notice to receive confirmation of quitting. Alternatively, you can quit DECKBUILD by using the **Define** menu. The appearance and location of the **Define** menu depends on the window manager in use, but is always available from the header region of the base frame. Under the **Open Look** window manager, click **MENU** with the pointer anywhere in the base frame header. Under the **Motif** window manager, click **SELECT** in the upper left-hand corner of the base frame header. In either case, activate the **Quit** option to quit from DECKBUILD.

If any unsaved edits exist, DECKBUILD displays a notice prompt allowing cancellation of the quit, or discarding of the unsaved edits and quitting. To save the edits, cancel the quit and save the file. You can either save to a new file or, if already editing a file, save edits back to the same file. To do this, select the corresponding option under the **File** menu. See Sections 4.18.1: “Text Subwindow Error Messages” and 4.18.2: “TTY Subwindow Error Messages” for more information.

It is often important to know whether there are unsaved edits, and exactly which file is being edited. DECKBUILD displays the file name in the header bar, or **NONE** if the file is not edited yet. When first saved to a file, it becomes the edited file. The word **edited** follows the file name if there are unsaved edits.

4.3: Invoking DeckBuild

4.3.1: Syntax

```
deckbuild [ -s3 | -an | -as | -od | -de | -fa | -ma | -hi |
-mo | -ut | -ss | -in ]
[-option [no]exec | [no]auto | [no]commands | [no]write] [-ascii]
[-simver <simulator_version>][-run][-outfile outfilename]
[-cutfile <cutfilename>] [-optfile <optfilename>]
[-remote <hostname>][-editfont <font>]
[-inpipe <input pipe> -outpipe <output pipe>] [-noplot] [-help]
[xview-arguments] [textedit-arguments] filename
```

4.3.2: Description

You can start DECKBUILD in either an interactive mode or a batch mode. In the interactive mode, it is possible to create, edit, and run input decks using mouse and keyboard operations. In the batch mode, DECKBUILD runs a previously created input deck. In the interactive mode, DECKBUILD appears as a window containing a text subwindow and a tty subwindow. The filename specifies the file to edit. If not specified, the text subwindow will be empty. If specified, DECKBUILD loads the file into the text subwindow.

In batch mode, a filename is required. DECKBUILD appears as an icon and automatically starts a simulator and executes the entire input deck (see “Default Simulator” on page 4-13). DECKBUILD quits when the run is complete. In either mode, you can save the run-time output of the simulation by specifying the `-outfile` option.

4.3.3: Options

The following options choose the default simulator configuration. A detailed explanation of these options is provided “Default Simulator” on page 4-13.

```
-s3 ssuprem3 -an athena -fa mercury
-od clever -asatlas -ma masksim
-de devedit-ututmost -mo mocasim
-ss smartspice-in internal -hi hipex
```

- **-run** starts DECKBUILD in batch mode. The input deck filename is required. If none is specified, DECKBUILD displays an error message and exits.
- **-outfile <outfilename>**— The file specified by outfilename is created to store the run-time output of the simulation. DECKBUILD writes each line of the tty subwindow to outfilename as the simulation progresses.
- **-cutfile <cutfilename>** — The file specified by cutfilename is loaded as the current cutline file. DECKBUILD uses the file for mask, region, and electrode data.
- **-optfile <optfilename>** can be used to load an optimizer data file with the input deck. The input deck must be specified and must match the optimizer data file. You can also use it with `-run` (synonym: `-opt`) to submit batch optimization runs.
- **xview/textedit-arguments** — DECKBUILD accepts standard OPEN LOOK and textedit command-line arguments. XView arguments can set such things as the color, font, and layout of DECKBUILD, while textedit arguments control such things as history limit, margins, and tab spacing.
- **filename** specifies the name of the input deck that DECKBUILD should initially load. If no filename is specified, DECKBUILD leaves the text subwindow empty. If the specified file does not exist, DECKBUILD displays a notice prompt to confirm its creation.

- **-option [no]exec | [no]auto | [no]commands | [no]write** toggles these **Main Control** options for a particular run. The options are `exec` for execute simulator, `auto` for auto interface, `commands` for the Commands menu, and `write` for write text to text window (all corresponding to the options setting on the Main Control popup). A `'no'` indicates to turn the option off. Repeat the `-option` argument to enable/disable more than one option.
- **-ascii** enables DECKBUILD to run in a non X windows environment. No popups or windows are created, but an input deck can be run normally. This option requires the use of an input filename and the `-run` option. The `-outfile` option (to store run-time output) is also helpful. If `-outfile` is not specified, the run-time output goes to `stdout`.
- **-simver <simulator_version>** specifies that the simulator should be started with the specified version. The simulator are invoked as `<simulator> -V <simulator_version>`. If no simulator is specified, the default simulator uses this version.

Note: `-sv` is also accepted as well as `-simver`.

- **-help** displays a list of the DECKBUILD and XView command line options.
- **-remote <hostname>** sets current simulations to be executed on the specified host. Although this option for interactive use can be used, it is intended for batch mode simulation. In interactive mode, you can select remote hosts from the **Simulator Properties** popup. See Section 4.15: “Remote Simulation” for more information.
- **-editfont <fontname>** allows the font for the text and tty sub windows to be specified as required.
- **-noplot** specifies that no plot commands within the deck are executed.
- **-inpipe <input pipe> -outpipe <output pipe>** specifies input and output pipe for communication with other products.

Examples

The following command will start DECKBUILD in interactive mode and pre-load the specified file.

```
deckbuild [filename.in] &
```

DECKBUILD can be submitted as a batch command on the UNIX command line. This method runs an input deck and quits at the end of the deck. You can submit a number of jobs for serial execution in this manner. The format of the command uses the `-run` option as follows:

```
deckbuild -an -run [filename.in]
```

DECKBUILD appears on the screen as a closed Icon and execute the named input deck, it may be opened to a full screen at any time during the execution. DECKBUILD exits completely when the last command in the input deck has been executed. If the runtime output is required to be stored into a separate file, the following options can be used:

```
deckbuild -an -run [filename.in] -outfile [filename]
```

Again, DECKBUILD appears as an Icon and executes the specified input deck. But in this case, all runtime output is appended to the named outfile.

If simulations are executed while X Windows is not running (or in a screen locked mode), the `-ascii` option can be used as follows:

```
deckbuild -an -run [filename.in] -outfile [filename] -ascii
```

DECKBUILD does appear as an Icon and executes the specified input deck. All runtime output is again appended to the named outfile, as recommended for `-ascii` use. If no outfile is specified for this type of command the runtime output is displayed in the current command tool.

Structure files saved during batch jobs should be carefully thought out. Make sure not to over write structure files with subsequent runs in the same working directory.

Defaults

A large number of control settings and options are configured at startup time. To configure these default options, use the **Save** function in the DECKBUILD property popups. DECKBUILD provides defaults saving routines on its property popups.

Default Simulator

At startup, DECKBUILD creates popups for the chosen default simulator and starts the default simulator in the tty subwindow if you enable the tty subwindow. DECKBUILD determines the default simulator according to the following rules:

1. The simulator specified on the command line, if any, takes precedence over all other rules.
2. If no simulator was specified on the command line, DECKBUILD uses the last saved default simulator. The default simulator is saved by clicking the **Save as Defaults** button on the Control Pad.
3. If no default simulator has been saved, DECKBUILD uses ATHENA.

Note: DeckBuild automatically changes simulators whenever it encounters an `autointerface` statement in the input deck (see “Autointerface” on page 4-2). Command line specification of the default simulator is important in the batch mode if there is no initial `autointerface` statement in the input deck.

4.4: DeckBuild Controls

DECKBUILD consists of a window (Figure 4-7) containing two subwindows: the text subwindow in the upper half of the base frame and the tty subwindow in the lower half. The text subwindow is used to build and edit input decks, while the tty subwindow is used to run the simulation. You can also display a messages subwindow below the tty subwindow.

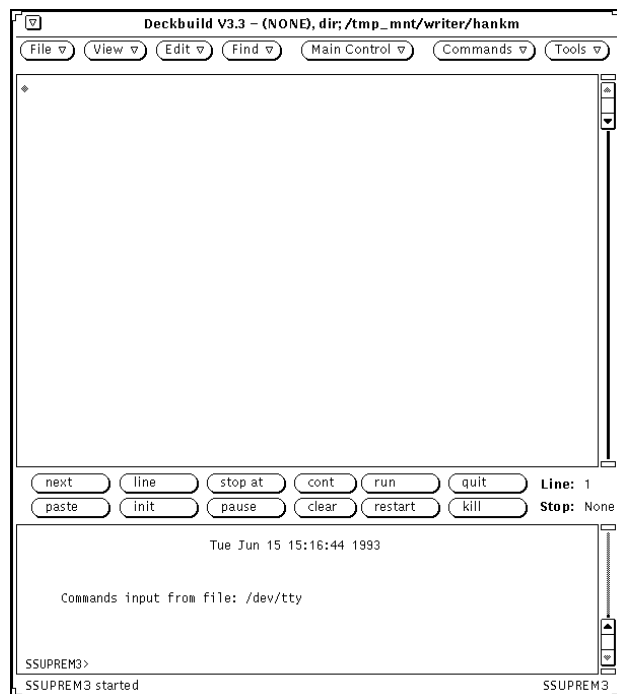


Figure 4-7: DeckBuild Base Window

4.4.1: Main Window Layout

DECKBUILD's controls are laid out in a hierarchical format. The most important and frequently-used controls are placed on the main window. Lower-level, less frequently used controls are stored as menu items on the top-level menus, or on popup windows accessed from the top-level windows. The resulting design attains an optimum of flexibility, ease of use, with a minimum of wasted space on the screen.

Menu Buttons

The set of buttons located along the top of the main window are the menu buttons. These buttons allow access to file control, simulator selection, simulator commands, and the tool palette and are as follows:

- **File** — The pulldown menu for saving and retrieving input deck files.
- **View** — The pulldown menu for changing the view of the text.
- **Edit** — The pulldown menu for cut, paste, and undo editing operations.
- **Find** — The pulldown menu for text search and replace operations.
- **Main Control** — The pulldown menu for top-level DECKBUILD configuration and control.
- **Commands** — The pulldown menu for simulator-specific commands and input deck creation.
- **Tools** — The pulldown menu for invoking the VWF INTERACTIVE TOOLS such as TONYPLOT.

Execution Control Buttons

The set of buttons located between the text and tty subwindows are the execution control buttons. These buttons allow complete interactive runtime control of the simulator and are as follows:

- **next** — This sends the current line to the simulator, and advances the current line by one. If a simulator is not running, one will be started.
- **line** — This resets the current line to the selected line.
- **stop at line** — This sets the breakpoint to the currently selected line.
- **stop now** — This stops execution after completing the current command and the associated history file's `save` command, if appropriate.
- **cont** — This continues the simulation from the current line to the end of deck, or to the breakpoint, if any. If not running, it also starts the simulator.
- **run** — This runs deck from the top to the bottom, or to the breakpoint, if any. If not running, it also starts the simulator.
- **quit** — This sends a `quit` statement to the simulator.
- **paste** — This sends the current selection to the simulator to be executed.
- **init** — If the selected text is a file, then the correct simulator is initialized with the file. Otherwise, the selected line is used to initialize from history.
- **pause/unpause** — This pauses/unpauses the running simulator.
- **clear** — This unsets the current breakpoint.
- **restart** — This restarts the current simulator if it's not running.
- **kill** — This kills the simulator.

4.4.2: Text & TTY Subwindows

Although the primary means of building and modifying input decks in DECKBUILD is via popup windows, DECKBUILD's text subwindow supports a full-featured text editor that allows editing decks directly. It is not necessary to know any special editing commands because they are available from the main window header menus. Command menus are accessed via the File, View, Edit, and Find menu buttons, and also by clicking MENU with the pointer anywhere in the text subwindow.

4.4.3: Using the Text Subwindow

Creating A New File

To store a new file, move the pointer to the File menu button and click and hold the **MENU** mouse button. Move the pointer to the **Save as...** menu item and release the **MENU** button. A **Save As** popup are displayed showing a list of directories and files in the current working directory (Figure 4-8).

To move between directories, either modify the **Directory** field and press RETURN or double-click on the required directory in the list. To save the required file, either highlight it in the list and select the **Save** button, double-click on the required file or enter a new file name in the **File** field and select the **Save** button.

Note: Saving a file to a new directory, moves DECKBUILD's current working directory to that location.

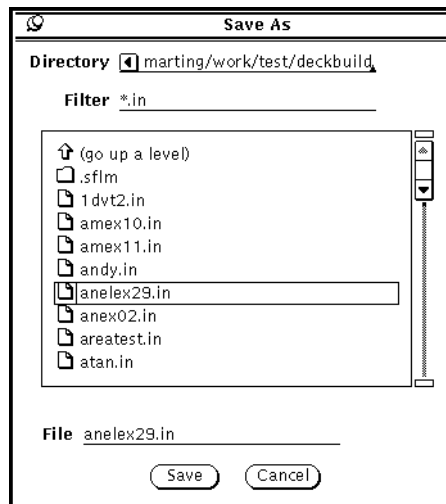


Figure 4-8: Save As Popup

Loading an Existing File

Any plain ASCII text file can be loaded into DECKBUILD. To load an existing file, choose the **Open...** item from the **File** menu. A file loader popup is displayed showing a list of directories and files, corresponding to the **Filter** field, in the current working directory (Figure 4-9). To move between directories, either modify the **Directory** field and press the Return key or double-click on the required directory in the list. To load the required file, either highlight it in the list and select the **Open** button or double-click on the required file.

Note: Loading a file from a new directory, moves DECKBUILD's current working directory to that location

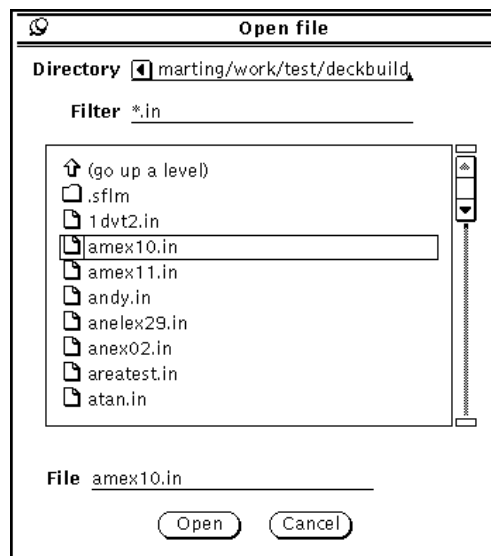


Figure 4-9: Open File Popup

Saving Changes

After changing the contents of an edited file (for example, by changing a diffusion parameter or by appending a device test to the end of a process deck), you can save the changes back to the same file or save the deck as a new file. To save the changes back to the same file, choose **Save** from the **File** menu. DECKBUILD saves the current file and makes a backup of the original renamed with a % suffix. For example, a file called `nmos.in` would be backed up to a file called `nmos.in%`. To save the changes to a new file, choose **Save as...** from the **File** menu. The procedure is the same as described above for saving a newly-created file.

Editing Input Decks

DECKBUILD allows you to search for text strings in a deck, copy text from one location and paste it to another, and add/delete text. To search for a text string, choose **Find and Replace** from the **Find** menu. The **Find and Replace** popup window appears with fields for text strings to find and to replace. Enter the text string to find on the first line. If desired, also type in the replacement text on the next line.

To search for each occurrence of a string, click **SELECT** on the **Find** button. Successive clicks find successive occurrences of the text. To replace a text string and approve each occurrence, click on the **Find** button to find each string. Then, click **Replace** to replace the text.

At each instance of the text string, if it is desired to replace the text and then search again, click **Replace** then **Find**.

To change only the first instance of a string without approval, click **Find** then **Replace**. To replace all instances of the text string immediately, choose **Replace All**.

Adding And Deleting Text

Add text to the deck by placing the insert point anywhere in the text subwindow (by clicking **SELECT** on the desired location) and click the **WRITE** button from any simulator pop-up window, or by simply entering the text. Delete text by choosing the insert point, then backspace over the text to delete.

Another way to insert and delete text is by using the system clipboard. To cut (that is, delete) text from the deck, click **SELECT** on the first character to cut. Then, click **ADJUST** on the last character to cut. This operation selects the range of text to cut. Finally, choose **Cut** from the **Edit** menu. The text is deleted from the input deck, and is placed in the system clipboard.

To paste (insert) text from the clipboard, place the insert point at the desired location, then choose **Paste** from the **Edit** menu. The text from the clipboard appears at the new location.

Copying Text

To copy a section of text, select the text to be copied (using one of the select mechanisms described above), then choose **Copy** from the **Edit** menu. This copies the selected text to the clipboard. To insert the text in the document, place the insert point at the desired location and choose **Paste**.

Tips On Cutting And Pasting

The following provides useful tips for cutting and pasting when working in the text Subwindow:

- Single-clicking **SELECT** in text selects a single character, double-clicking selects a word, triple-clicking selects a line, and clicking four times selects the entire document.
- Some may prefer the **Cut**, **Paste**, and **Copy** buttons on the keyboard (not available on all keyboards) rather than using the **Edit** menu.
- Text copied to the clipboard remains there until another copy operation is done.
- Text may be copied to the clipboard from one application, and pasted into an entirely different application.
- DECKBUILD's tty subwindow also supports cut and paste operations.

4.4.4: Using the TTY Subwindow

The tty subwindow, used to drive the simulation programs, is a text-based command window. It accepts many of the same commands and has many of the same capabilities as the text editor.

Entering Commands Directly

The execution control buttons provide the primary means of input to the running simulator, but text typed directly into the tty subwindow can be used as well. Place the pointer anywhere in the tty subwindow and click on **SELECT**. If a simulator is running and is waiting for input, a caret appears at the simulator prompt. Commands typed at the keyboard are executed by the simulator.

The text of the current command line can be edited using the normal text editing functions explained above.

Editing The Contents

Except for the current command line, all lines in the tty subwindow are read-only and cannot be edited. The subwindow displays a log of the simulator input and output, which can be scrolled using the scrollbar. To search for a particular text string, use the **Find** and **Replace** popup. Bring up the tty (Term Pane) menu by placing the pointer anywhere in the tty subwindow and click on **MENU**. This menu contains a number of items used to edit and manipulate the tty subwindow, and is like the text menu. Choose **Find and Replace** from under the **Find** menu item. Searching is done the same as with the text subwindow.

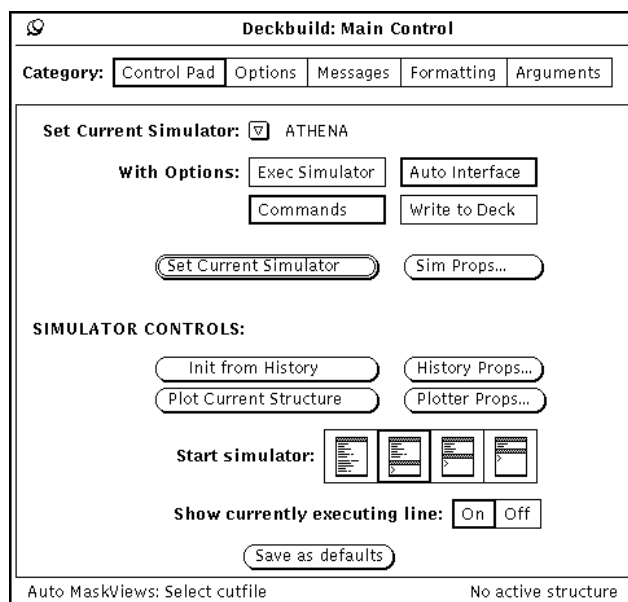


Figure 4-10: Main Control Popup

Cut, Paste, And Copy

You can copy and paste text in the tty subwindow directly by using the **Edit** menu from the tty menu or by using the corresponding keys if available. Cutting is not allowed on any but the current command line, however, since the remainder of the subwindow contents are read-only.

Saving/Resetting The Contents

To reset the contents of the `tty` subwindow (that is, remove the log and clear the subwindow), bring up the `tty` menu and choose **History→Clear log** (Figure 4-10). To save the contents of the `tty` subwindow, bring up the `tty` menu and choose **History→Store log as new file**.

Note: **History** on this menu refers only to the contents of the `tty` subwindow, and is not related to DECKBUILD's **History** feature.

Error Messages

The Error Messages section at the end of this chapter contains information about error messages that may be encountered while using the text and `tty` subwindows.

4.5: Main Control

The **Main Control** popup (Figure 4-10) provides a collection of controls, organized by category, that allows customizing of DECKBUILD's configuration.

The **Main Control** popup with the **Category** menu displayed is shown in Figure 4-9. The **Category** menu contains the following control category selections.

- **Control Pad** — Simulator choice, activation, and runtime options.
- **Options** — User configurable option settings.
- **Messages** — DECKBUILD debugging control.
- **Formatting** — Input deck operation formatting.
- **Arguments** — Command line arguments for other VWF INTERACTIVE TOOLS.

Choose a category by clicking on **MENU** with the pointer over **Category**. Move the pointer over the category of interest, then release the **MENU** button. When you select a new category, the popup changes from the old category to the new. Old controls disappear, new ones appear, and the popup may change its size.

Each category contains a **Save as defaults** button. Click on the button to save the settings in that category. DECKBUILD configures itself with the saved settings the next time it is invoked.

4.5.1: Control Pad

The **Control Pad** provides one-stop service for the highest level of DECKBUILD configuration and runtime control. Choose the currently configured simulator, auto interface options, and DECKBUILD layout using the **Control Pad**. In addition, the important runtime control features of **History** and **Plot** are placed here for easy and quick use.

4.5.2: Choosing a Simulator

Choose a new simulator when writing or modifying a deck for that simulator, to see its **Commands** menu, or to shut down the currently running simulator and start up the new simulator. DECKBUILD provides control over each of these actions with **Options**. To change over to a new simulator, click and hold **MENU** in the **Set Current Simulator** setting. A list of available simulators is shown (Figure 4-11). Move the pointer over the desired simulator and release the **MENU** button. Activate the options that are appropriate (usually all except **Write to Deck**). Finally, click on the **Set Current Simulator** button.

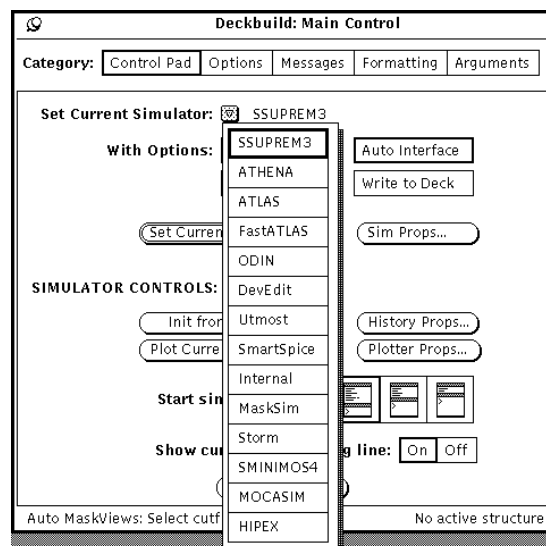


Figure 4-11: Main Control - Set Current Simulator

In the following section, the current simulator means the simulator before the button is clicked, and the new simulator means the current simulator after the button is clicked. Based on the chosen options, the following occurs:

1. **Exec Simulator** causes the currently running simulator to shut down and the new simulator to start. If you enable the **Auto Interface** option and DECKBUILD determines that an interface is appropriate, then DECKBUILD performs an auto interface from the current to new simulator. **Exec Simulator** is enabled by default.
2. **Auto Interface** causes an auto interface to be performed between the current and new simulator. This option is also referenced at run time whenever DECKBUILD encounters an auto interface statement in the input deck. See Section 4.10: “Auto Interfacing” for more information. **Auto Interface** is enabled by default.
3. **Commands** causes DECKBUILD to change the **Commands** pull-down menu to reflect the new simulator’s syntax. All popups for the current simulator that are not pinned are closed. **Commands** is enabled by default.
4. **Write to Deck** causes an auto interface statement to be inserted in the input deck at the location of the text caret. See Section 4.16.4: “AUTOELECTRODE” for more information. **Write to Deck** is not enabled by default.

Note: DECKBUILD always delays popup creation until the popups are needed to reduce startup time. The **Commands** menu and associated popup windows are created only when referenced for the first time. Therefore, if the new simulator has not been referenced before, then there is a short delay while the popups are created. Otherwise, the change will be instantaneous. A notice in the lower left footer of the main window will be displayed while the popups are being created.

When the change from one simulator to another is complete, the new current simulator is shown in the lower right footer of the base window. The current simulator is always shown in the lower right footer.

4.5.3: Simulator Properties

To access properties unique to each simulator, click on the **Sim Props** button. A popup is displayed that contains settings and options applicable to the current simulator, including a switch for local and remote Simulation and command-line arguments used to execute the simulator. If remote simulation is selected, the hostname of the machine that executes the simulator must be specified. See Section 4.15: “Remote Simulation” for more information.

4.5.4: Start Simulator

The **Start Simulator** setting determines DECKBUILD’s window layout. Each of the four choices in the setting has an iconical representation of what DECKBUILD can look like: text subwindow only, or text subwindow with small, medium, or large tty subwindow displayed. By default, DECKBUILD appears with the tty subwindow displayed in the small configuration.

The current simulator is started when the tty subwindow is enabled. From that point on, the simulator continues to run, even if the tty subwindow is made to disappear.

4.5.5: Simulator Controls

Three high-level run-time functions are provided on the **Control Pad: Init from History, Plot Current Structure, and Show Currently Executing Line**.

Init from History is used to re-initialize the simulator from some previously-run line in the input deck. DECKBUILD automatically saves files as each line in the deck is run. This permits transparent movement back and forth in the input deck if you need to back up to try something again. **History** properties, including the ability to disable history, are accessed by clicking on the **History Props...** button. See Section 4.9: “History” for more information.

Plot Current Structure is used to plot either the current structure or any selected structure. It provides a shortcut to **Plot** on the **Tools** menu. The following occurs when you click on this button.

- If there is text selected (highlighted) anywhere on the screen, DECKBUILD takes the text as filename of a file to plot. DECKBUILD starts up TONYPLOT on the named file.
- If no text is selected and a simulator is running, DECKBUILD causes the simulator to save its simulation data and then starts TONYPLOT using that data. This does not disrupt any lines of input deck waiting to be executed by the simulator.
- Optionally, choose a set file by clicking on the **Plotter Props...** button. The set file is used to record a given plot's layout, such as scaling, zoom, number and type of plots shown. After creating a set file, you can use it to re-create the same layout when using the same or any other plot data. A set file is often useful for comparing the results of different simulation runs.

After pressing the **Plotter Props...** button, the **Plotter Set Files** popup will appear. This contains a scrolling list of set files in the current directory, and a text fields used to search for set files. Adjust the directory name and directory filter if necessary. Click **SELECT** over the name of the desired set file, if any, in the scrolling list. If none are desired, then make sure no entries are selected (de-select a selected list entry by clicking **SELECT** on it again). The selected entry, if any, will be used as the set file on subsequent plots.

It is also possible to specify when DECKBUILD should save the active structure (no filename highlighted). See the **Plot structure** description in Section 4.5.6: "Options Category". It provides a shortcut to **Plot** on the **Tools** menu.

Show Currently Executing Line tells DECKBUILD whether or not to select (highlight) each line in the input deck as it is executed by the simulator. DECKBUILD also automatically scrolls the text subwindow to keep the currently executing line always in view. This feature is enabled by default, except when running batch mode (`-run`).

4.5.6: Options Category

The **Options** category (Figure 4-12) is a collection of a number of settings that modify the behavior of various DECKBUILD functions. For example, it is possible to configure what happens when **WRITE** is clicked on a syntax popup, whether history files should be automatically removed at the end of a run, and even the nice value of the simulator.

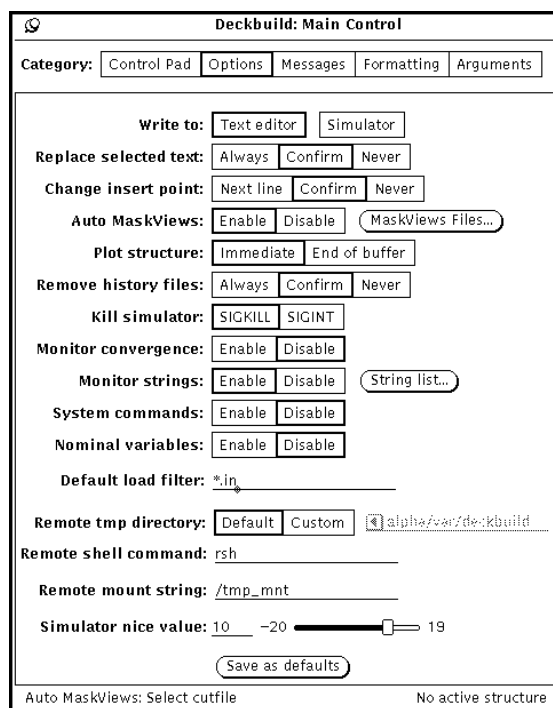


Figure 4-12: Main Control - Options Category

Write to setting determines what happens when **WRITE** is clicked on any of the syntax popups. It allows choosing to send the text to either the text subwindow or directly to the simulator running in the tty subwindow or both. The default is **Text Editor** only.

Replace selected text tells DECKBUILD what to do when **WRITE** is clicked on while any line of text is selected (highlighted) in the text subwindow. This situation often comes about after parsing a line of the deck, see Section 4.7.2: “Parsing the Deck”. DECKBUILD provides the choice of replacing the selected text with no confirmation, confirming each replacement, or not allowing replacement at all. If you activate **Confirm**, then DECKBUILD displays a notice prompt to either confirm or cancel the overwrite each time. If you activate **Never**, DECKBUILD displays a notice prompt confirming that the operation was cancelled. The default is **Confirm**.

Change insert point modifies the **WRITE** insertion behavior. This feature is designed to make deck building easier by checking the location of the text caret each time **WRITE** is clicked on. The caret should be at the beginning of a line. It could be relocated for various editing purposes, resulting in the cursor being left in the middle of a line. **Next Line** automatically moves the caret to the beginning of the next line. **Confirm** asks for confirmation or cancellation of the move each time. If confirmed, DECKBUILD moves the caret to the next line. **Never** inserts text wherever the cursor is located. The default is **Confirm**.

Auto MaskViews enables or disables the automatic substitution of MASKVIEWS layout information during deck execution. DECKBUILD also displays the status of this choice in the left footer of the **Main Control** popup for easy reference during run time. The **MaskViews Files...** button is placed here as a convenient way to select cut files (also accessible from the **Cut files...** under MASKVIEWS on the Tools

menu). See Section 4.16.11: “MASKVIEWS” for a complete description on how to use it with DECKBUILD. The default is **Enable**.

Plot structure controls where interactive plots are made. Interactive plots are done from either the **Tools** menu or from the **Control Pad**. Normally, DECKBUILD saves and plots the active structure right away. The simulator, however, may be busy executing several lines from the input deck. This can happen, for example, after clicking on the **run** button, which causes DECKBUILD to queue up many lines from the deck to be run and feeds them down one at a time. In this case, it is possible to define whether the `save` and `plot` commands are placed at the beginning or at the end of those queued commands. Choose either **Immediate** or **End of Buffer**. The default is **Immediate**.

Remove history files defines what DECKBUILD does with history files when DECKBUILD exits. Choosing **Always** causes DECKBUILD to always clean up history files after itself. **Confirm** brings up a confirmation notice prompt when DECKBUILD exits. You can then confirm or cancel history file removal. **Never** ignores history files and does not elicit a notice prompt. This is useful if you want to modify the history files on a regular basis.

Note: It's OK when history files are not removed as long as there's enough disk space. They'll eventually start getting re-used, so they won't pile up endlessly. See Section 4.9: “History” for more information. The default is **Confirm**.

Kill simulator determines how DECKBUILD kills the simulator when the **kill** button is clicked on. `SIGKILL` is guaranteed to kill the simulator, but it does not give the simulator a chance to clean up after itself. If the simulator uses any temporary files when killed, the temporary files will not be removed and will remain in the current directory, or in `/tmp`. It uses the Unix “kill” signal 9. On the other hand, `SIGINT` allows the simulator to remove its temporary files and perform any other required actions before it terminates. It uses the Unix “interrupt” signal 2. We recommend that you use `SIGINT` for normal use. If the simulator does not seem to die properly under some circumstances, switch to the forced kill of `SIGKILL`.

Monitor convergence, if enabled, monitors the output from ATLAS and looks for messages indicating that the simulator has failed to converge. If convergence failure is detected, an error message is displayed and the simulation is halted. No more lines from the input deck are sent to the simulator, and the simulator is left running and displaying its interactive prompt. You should enable this option to stop ATLAS runs at the first sign of convergence failure. You should disable this option for doing snapback characteristic simulation, which actually depends on convergence failure as a precondition to switching boundary conditions and continuing the simulation.

Monitor Strings, if enabled, monitors the TTY output for strings selected in the **Monitor String List** (Figure 4-13). If a selected string is detected, a message is displayed and the current simulation is stopped at this point. Note the simulator stays active.

System Commands, if enabled, allows DECKBUILD to execute UNIX system commands within a simulation deck. To use a system command, the line must start with the command system as shown below.

```
system rm.history*.str
```

To enable system commands for VWF AUTOMATION TOOLS, set the environment variable `DB_SYSTEM_OPTION` to any value.

Nominal Variables, if enabled, allows you to use the nominal flag for DECKBUILD set variables. If you specify the nominal flag in a `set` statement, the variable remains unchanged if already in existence. The variable will be created as specified if it is new. See Section 4.16.12: “SET” for more details on the `set` command.

Default Load Filter sets the default file filter for DECKBUILD's file loader popup.

Remote options are used to perform remote simulation. These options are explained in Section 4.15: “Remote Simulation”.

Simulator nice value sets the priority of the simulator process. Negative nice values give the simulator process more CPU time relative to other processes. Positive nice values give it less. Only the super-user can use negative values. The value is used only when the simulator is started and does not change the nice value of the simulator once it is running. You must then quit and restart the simulator to give it a new nice value. The default value is 10 (added).

To set the simulator nice value for VWF AUTOMATION TOOLS, set the environment variable `NICE_ARG` to the required number.

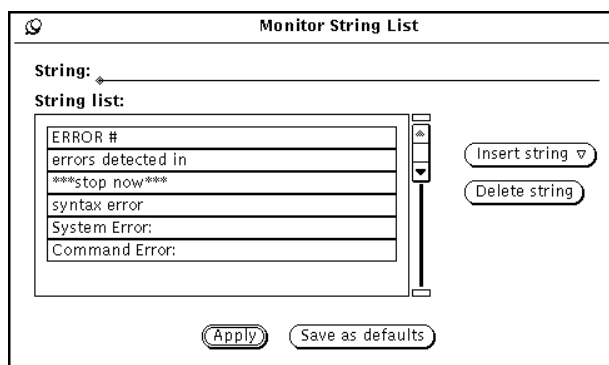


Figure 4-13: Monitor String List Popup

4.5.7: Messages Category

The **Messages** category (Figure 4-14) contains option settings that control the display and behavior of DECKBUILD’s messages window.

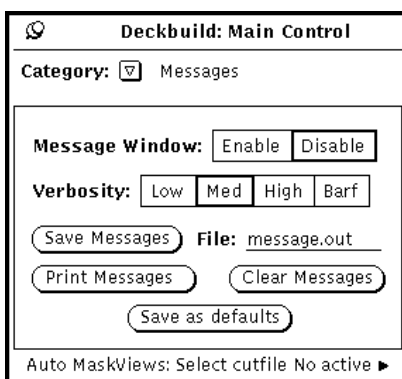


Figure 4-14: Main Control - Messages Category

The **Message** Window provides a debugging log of various actions that DECKBUILD takes as it runs, such as input decks and switches between simulators. The **Message** Window is useful to track user activity in the input deck since activity messages are all time-stamped. This times how long various sections of the input deck take to execute.

The window is enabled with the **Message Window** setting. When enabled, the window appears as a small, read-only text window at the bottom of the main window, and below the tty subwindow.

Messages are logged as a function of the selected **Verbosity** level. A general rule of thumb is as follows:

- **Low** — Tracks major events, such as simulator starts, stops, and simulator switching. Also, many warning messages that appear on the screen are logged at this level.

- **Med** — Low plus auto interfacing information. This is the default setting.
- **High** — Med plus input deck lines as they are queued to be run.
- **Barf** — High plus input deck lines as they are actually executed.

4.5.8: Formatting Category

The **Formatting** category (Figure 4-15) controls the format and initial state of input deck operations, and is used in conjunction with the VWF.

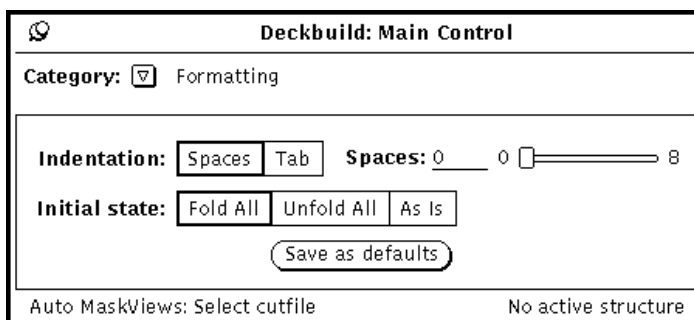


Figure 4-15: Formatting Category

Indentation sets the number of spaces that each level of operations is indented. Top-level operations are flush with the left margin, while operations they contain are indented by this amount and so on with each level of nested operations.

Initial state controls what the format of the deck will be when DECKBUILD is invoked from the VWF. Operations can either be completely folded so that only top-level operations appear, or completely unfolded so that all operations appear in their entirety. The deck can also be left as-is in the state when it was saved to the database.

4.5.9: Arguments Category

The **Arguments** category (Figure 4-16) sets the default command-line arguments used by DECKBUILD to start certain other VWF INTERACTIVE TOOLS.

To change command-line arguments, or to use a different program, enter the new arguments and click on the Return key. The new arguments are used the next time that program is started.

DECKBUILD automatically appends specific arguments to the ones that were entered. For example, DECKBUILD appends a filename to the **Plotter** argument whenever the current structure is plotted, and appends a layout filename when MASKVIEWS is started. For this reason, it is unnecessary (or wise) to put specific filenames in the argument lists.

When encountering a `tonyplot` statement during execution, DECKBUILD also automatically determines if a structure is 3D and use the appropriate **3D plotter** argument.

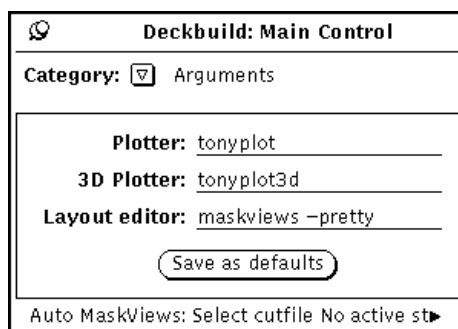


Figure 4-16: Main Control Arguments

4.6: Execution Control

The execution control buttons, grouped between the text and tty subwindows when the tty subwindow is enabled, control how the input deck is sent to the simulator. It is possible to step through the deck a line at a time, run up to a point, run the entire deck, or even back up and continue. DECKBUILD provides the features needed to start, run, and quit the simulation.

4.6.1: Execution Concepts

DECKBUILD always remembers the current line in the input deck. The current line is the line that is next sent to the simulator, and is always shown next to **Line** on the right side of the control panel. It is updated automatically when stepping through or running the input deck, initialize from a filename, re-initialize from history, or when explicitly reset.

DECKBUILD sends lines one at a time to the simulator when the simulator is ready for more input. Lines that have been sent to the simulator (using the **next** or **run** buttons, for example) are buffered until the simulator is ready to accept them. Therefore, you can send down an arbitrary number of lines to be simulated all in one go, and the lines will be executed in the order received. The buffer is cleared automatically whenever the simulator exits. See Figure 4-17 for a view of the **Execution Control** buttons.

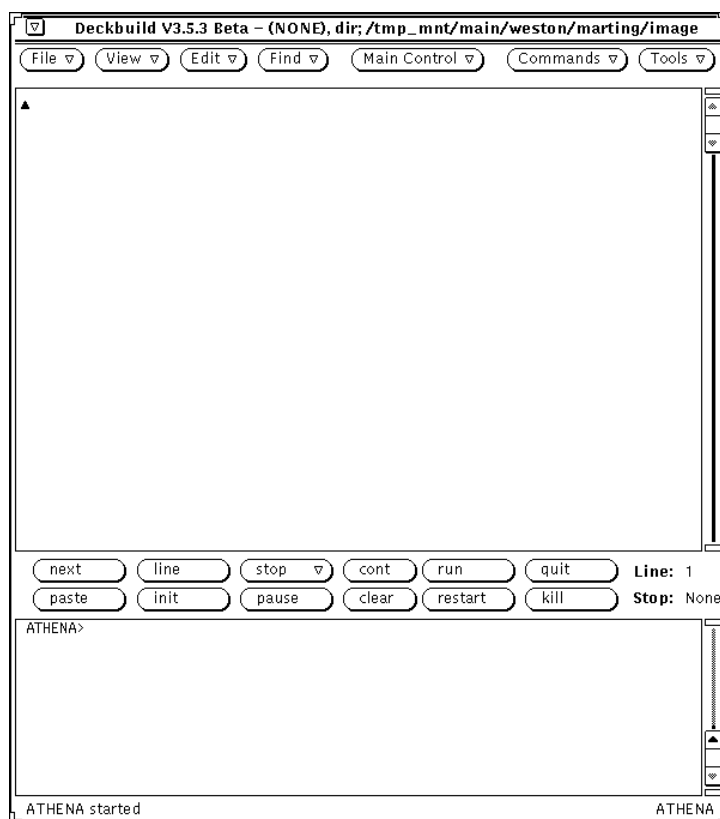


Figure 4-17: The Execution Control Buttons

4.6.2: Execution Control Buttons

The following describes the execution control buttons.

- **next** — Sends the current line to the simulator and advances the current line by one.
- **line** — Resets the current line to the currently selected line.
- **stop at line** — Sets the breakpoint to the currently selected line.
- **stop now** — Stops current execution after completing the current command and the associated history file's save command, if appropriate.
- **cont** — Continues the simulation from the current line to the end of deck or to the breakpoint, if any.
- **run** — Runs the deck from the first to the last line or to the breakpoint, if any.
- **quit** — Sends a quit statement to the simulator.
- **paste** — Sends the current selection to the simulator to be executed. The current selection may exist in any application, or in DECKBUILD itself. Use **paste**, for example, to paste in lines from another input deck that might have been viewed using a text editor.
- **init** — If the selected text is a file, then the correct simulator is initialized with the file. Otherwise, the selected line is used to initialize from history.
- **pause/unpause** — Pauses/Unpauses the simulator.
- **clear** — Unsets the current breakpoint.
- **restart** — Restarts the current simulator if it is not running.
- **kill** — Kills the running simulator.

4.6.3: Stepping Through and Running the Deck

You can execute one line at a time by pressing the **next** button. The current line is sent to the simulator, and the text caret moves to the next line. Press **next** again to execute the next line. Continue stepping lines through all or part of the deck.

Alternately, run the whole deck by pressing **run**. The deck is executed from the first line to the last. Press **cont** to continue onwards from the current line all the way to the end of the deck. Both **run** and **cont** stop at the breakpoint if one is set.

You can stop the execution at any point by pressing the **now** option on the **stop** menu. This does not quit the simulator, but halts the execution after completing the current command, along with associated history file save if appropriate.

4.6.4: Setting and Clearing Breakpoints

If you want to run the input deck only up to a certain line, set a breakpoint on that line. DECKBUILD runs up to, but not including, the breakpoint. Set the breakpoint by selecting (highlighting) all or any part of the desired line, then click on the **at line** option on the **stop** menu. The breakpoint is displayed as **Stop** on the right side of the execution panel.

Clear the breakpoint by clicking on the **clear** button.

If multiple breakpoints are required in a deck, the use of the **Monitor Strings** option stops the execution at locations marked by selected comments. For example, if the comment `# stop here` were entered at different positions in an input deck and this string was then added to the enabled **Monitor Strings** list, the simulation would stop at each location with a message.

4.6.5: Setting the Current Line

As already mentioned, the current line is automatically maintained as the deck is stepped through and run. DECKBUILD places the text caret on the current line each time the line is reset. You can set the current line anywhere by selecting (highlighting) any part of the desired line and clicking on the **line** button. The **Line** display is then updated to reflect the change.

4.6.6: Pausing, Stopping, and Restarting the Simulator

Click on the **quit** button to send a **quit** command down to the simulator. If there are no buffered commands, the quit is executed immediately, and the simulator exits. DECKBUILD knows when the simulation will end and will print out the following message to the tty subwindow.

```
***END***
```

In this case, you may want to use **kill**, which kills the simulator immediately, or the **stop now** button to stop the execution after the current command. The **stop now** function does not kill the simulator so you can continue by using the **cont** button.

To restart the simulator, click on **restart**. The current simulator is then started. To start a different simulator, choose the new simulator as described under **Main Control**. The **Set Current Simulator** button on the **Control Pad** always starts the current simulator if the **Exec Simulator** option is set.

You can also pause the simulation. Pausing is equivalent to typing a control-Z in the C-Shell: the simulation immediately relinquishes its use of the CPU and stops. Unpausing the simulator causes it to continue from exactly where it left off, and is equivalent to bringing a paused job back to the foreground in C-Shell.

Pausing is frequently used to free up some CPU cycles for some other temporary task. When the task is finished, the simulation can be unpaused. Pause the simulator by clicking on **pause**. The button's title changes to **unpause**. Click on the button once again to unpause the simulator.

4.6.7: Initializing the Simulator

DECKBUILD provides a shortcut to initialize the simulator. Select the name of a structure file to initialize and press **init**. DECKBUILD takes the selected text as a filename, figures out which simulator the file came from, starts that simulator, and executes the proper INITIALIZE statement.

For example, if you were running a ATHENA simulation and saved the structure file ATHENA.str, you can restart ATHENA (if it's not already running) and initialize it with that structure by selecting the text ATHENA.str anywhere on the screen and clicking on **init**. DECKBUILD then automatically starts ATHENA and executes the statement INIT INFILE=ATHENA.str. Note that the file may exist in another directory. If so, it's necessary to form the file name properly. If ATHENA.str is in /usr/jdoe/an and running DECKBUILD from /usr, select the file name /usr/jdoe/an/ATHENA.str or jdoe/an/ATHENA.str.

Note: Using the **init** button while selecting a line in the deck, instead of a structure file, will result in initialization from history files, assuming history files are present for the selected line.

4.7: Commands

The primary means of accessing popups that are used to write the input deck is the **Commands** menu. Typically, each item on the menu is associated with a popup window that contains controls used to specify an input deck command. For instance, invoking **Implant...** under ATHENA causes the ATHENA Implant popup to appear.

There are different **Commands** menu for each simulator. The menu always reflects the current simulator shown in the lower right-hand corner of the main frame. Changing the current simulator will cause a different menu to appear as described in the Section 4.4: “DeckBuild Controls”.

4.7.1: Deck Writing Paradigm

In general, DECKBUILD uses one of two ways to write an input deck: either all at once or line-by-line. DECKBUILD uses each as appropriate. Process simulation, for example, is an inherently sequential operation. The same basic commands (`implant`, `diffuse`, `etch`, and `deposit`) are used over and over again. SSUPREM3 and ATHENA are good examples of how this paradigm works, because each popup has a button used to just write the syntax for that popup/command. On the other hand, the interface to MERCURY is represented as device specification and a set of solutions/tests that are applied to that device. In this interface, click on a single button to write a deck.

4.7.2: Parsing the Deck

DECKBUILD has a built-in feature that allows parsing any part of a deck to automatically configure the appropriate popup(s). For example, to repeat a previous `implant` process command with some minor changes, parse the `implant` statement. To do this, reset the controls of interest on the Implant popup, place the text caret in the proper location, and press the **WRITE** button. To parse any fragment of text, highlight the text and select **Commands**→**Parse Deck...** DECKBUILD scans the highlighted text, determines the proper popups to change, resets the settings of all parameters specified in the highlighted text, and makes the popup(s) visible.

A Few Points to Observe

- **Parse Deck** does not change the settings of parameters that are not specified. If parsing the line `implant boron`, the values of energy and dose, for example, will not be altered from whatever previous value they had on the popup.
- **Parse Deck** parses any highlighted text whether it is in DECKBUILD’s own text subwindow or in a separate program. If in DECKBUILD’s text subwindow (the usual case), DECKBUILD automatically extends the selection of a partial line to cover a full line.
- Highlight as much text, covering as many command statements, as desired. DECKBUILD configures and brings up all appropriate popups for the current simulator. If you highlight more than one of the same statement, the last has priority.
- DECKBUILD ignores all text that it does not understand.

4.7.3: Process Simulators

Figure 4-18 shows the **Commands** menu for a process simulator (ATHENA).

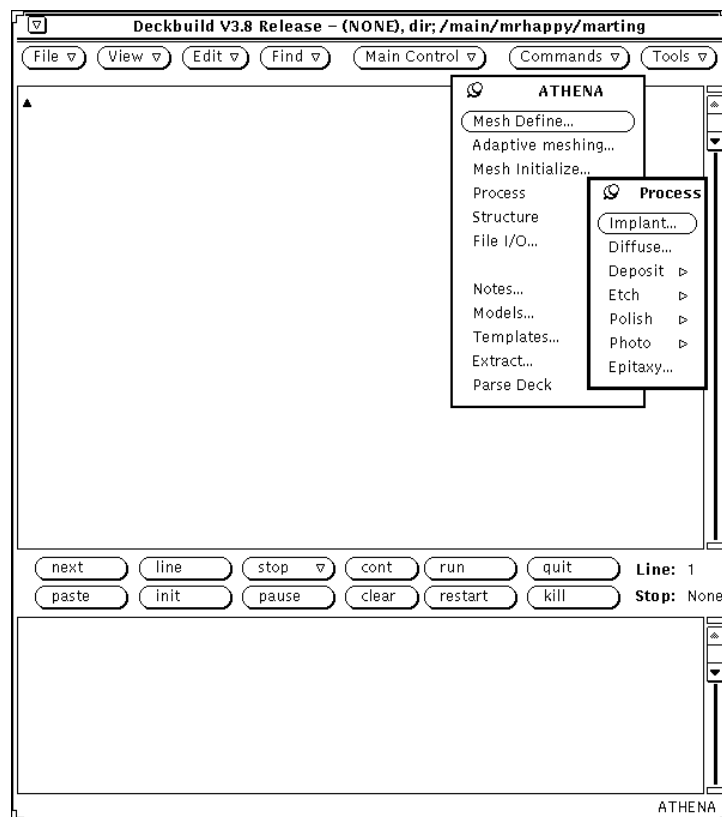


Figure 4-18: Command Menu of ATHENA

Writing a Process Input Deck

Since process fabrication is itself an inherently sequential operation, simply choose the command of interest from the **Commands** menu. A corresponding popup appears that has controls laid out to represent the variable parameters available for the command. For example, Figure 4-19 shows the ATHENA Diffusion popup.

Deckbuild: ATHENA Diffuse

Display: **Time/Temp** Ambient Impurities Models settings

Time/temperature:

Time (minutes): 30 0 500

Temperature (C): 1000 800 1300

End temperature (C): 1300 800 1300

Temperature rate (C/min): 0.000 Rate: Variable

Temp: Constant Ramped

Ambient:

Ambient: Dry O2 Wet O2 **Nitrogen** Gas Flow

Gas pressure (atm): 1.00 0.00 10.00

HCL %: 0 10

Comment: _____

WRITE Properties ▾

Figure 4-19: ATHENA Diffusion Popup

Selecting the Categories

Some popups, such as Figure 4-19, contain a non-exclusive **Display** setting at the top of the popup in an attempt to conserve valuable screen space. Click **SELECT** in the boxes to display/undisplay the setting of interest. When enabling a setting such as **Impurities**, the entire popup grows vertically to hold the new section. The popup shrinks again when the setting is disabled. Select as many or as few boxes as needed.

Writing the Text

When all of the controls have been adjusted to reflect the process step to be performed, click the **WRITE** button. A line (or sometimes lines) of text is written to the deck at the location of the text caret. If desired, verify the caret's location before clicking **WRITE**, although DECKBUILD automatically detects if the caret is in the middle of a line and moves it if necessary. For more information, see Section 4.5: "Main Control".

Build the entire process deck by invoking the popups as needed from the **Commands** menu, setting the controls, and writing the deck a popup at a time. You can also parse the deck. That is, read a line or lines of syntax from the deck and automatically configure the correct popup(s). For more information, see Section 4.7.2: "Parsing the Deck".

4.7.4: Mercury Tool

For the MERCURY simulator, the **Commands** menu provides a single button to start the external MERCURY TOOL application. This interface tool provides the ability to create and modify MERCURY simulation decks with ease. For more information about this interface, see the MERCURY USER'S MANUAL.

4.7.5: Clever

DECKBUILD also provides a set of command popups for CLEVER and EXACT products. For more information, see the CLEVER or EXACT USER'S MANUALS.

4.8: Tools

DECKBUILD's Tools menu provides the interface to other VWF INTERACTIVE TOOLS: TONYPLOT, MASKVIEWS, and MANAGER (Figure 4-20). In addition, there is a general **Text Editor** for viewing other files such as external simulation decks executed by a **source** statement (see Section 4.16.13: "SOURCE") from within the current deck.

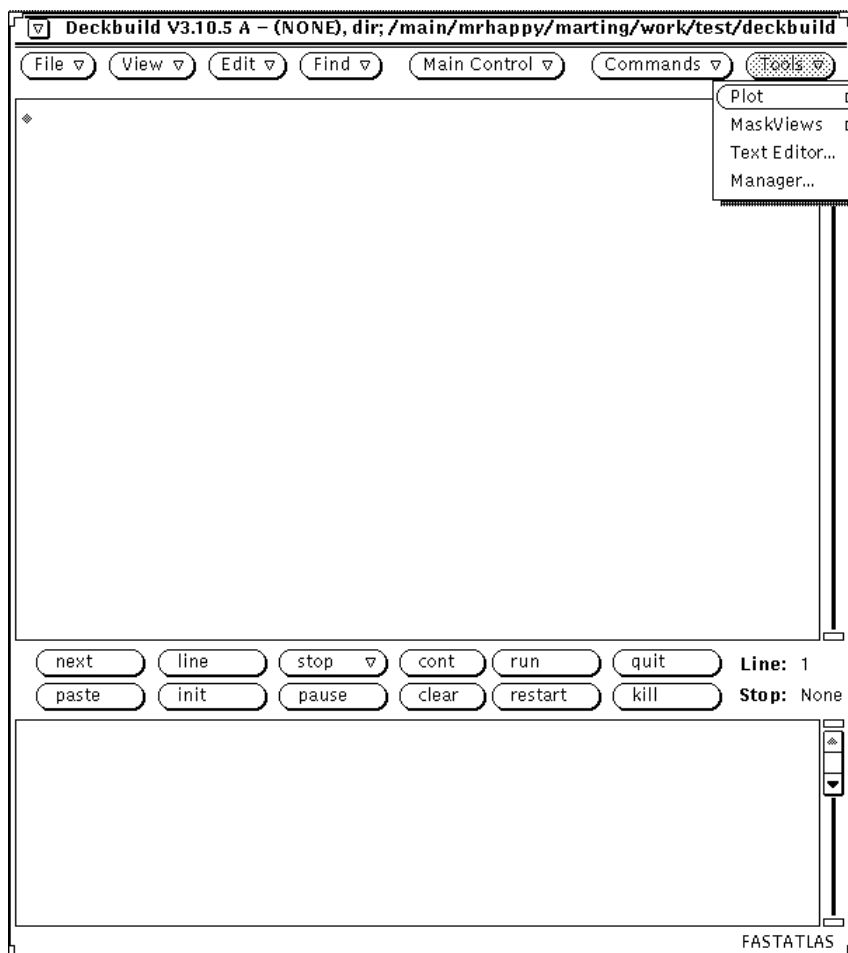


Figure 4-20: The Tools Menu

4.8.1: Starting TonyPlot

Plot simulation results from DECKBUILD by choosing the **Plot structure...** choice in the **Plot** pull-right menu. DECKBUILD allows either plotting the current structure or any specified structure. The following rules are:

- If there is text selected (highlighted) anywhere on the screen, DECKBUILD takes the text as the name of a file to plot. DECKBUILD starts up TONYPLOT on the named file.
- DECKBUILD automatically determines if the selected file is 3D and starts up TONYPLOT3D if appropriate.
- If no text is selected and a simulator is running, DECKBUILD causes the simulator to save its simulation data, then starts TONYPLOT on that data. This does not disrupt any lines from the input deck waiting to be executed by the simulator. For CLEVER, which saves a structure, log file and a layout, two TONYPLOTS and a MASKVIEWS are invoked. Optionally, choose a set file by activating the **Set files...** choice.

Note: The set file is used to record a given plot's layout, such as scaling, zoom, number, and type of plots shown. After a set file is created, it can be used to re-create the same layout when using the same or any other plot data, and often useful for comparing the results of different simulation runs.

The **Plotter Set Files** popup appears (Figure 4-21). This contains a scrolling list of set files in the current directory, and a text field used to search for set files. Adjust the directory name and directory filter if necessary. Click **SELECT** over the name of the desired set file in the scrolling list. If none are desired, ensure that no entries are selected (de-select a selected list entry by clicking **SELECT** on it again). If an entry is selected, it is used as the set file on subsequent plots.

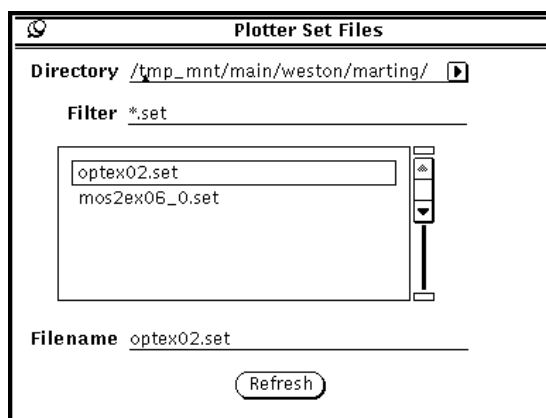


Figure 4-21: Plotter Set Files Popup

Since **Plot structure...** is the default **Tools** menu item, simply click **SELECT** on **Tools** to activate it. This is easier than descending through two levels of menus. Also see **Plot structure** under **Options** on the **Main Control** popup to determine when interactive plots will be made if many lines from the input deck are waiting to be simulated. An option to plot can be set immediately, or at the end of the simulation. The default is immediate.

4.8.2: Starting Maskviews

DECKBUILD allows you to bring up MASKVIEWS with an optional layout file. To choose the layout file, select **Start MaskViews...** from the MASKVIEWS pull-right menu. The MASKVIEWS **Layout Files** popup appears (Figure 4-22). This contains a scrolling list of layout files in the current directory and text fields to search for layout files. Find and select the layout file of choice, then click on the **Start MaskViews** button on the popup (Figure 4-23). After a few moments, MASKVIEWS appears with the specified layout file loaded. If you did not choose a layout file, MASKVIEWS starts with no layout file loaded.

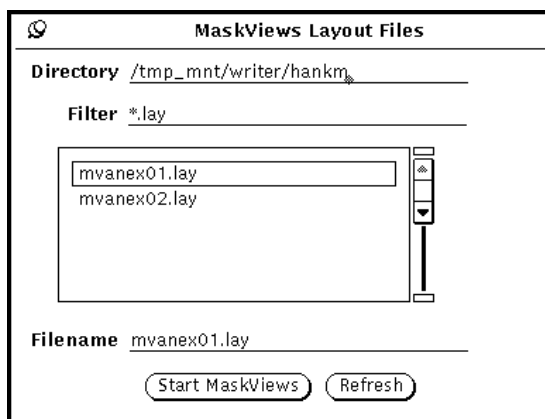


Figure 4-22: MaskViews Layout Files Popup

Loading a Cutline

There are two ways to load a MASKVIEWS cutline from the **MaskViews Cut Files** popup and an alternative method at run time in the simulation deck. To load from the popup, select **Tools→MaskViews→Cut files...** and either save a file from MASKVIEWS and load it into DECKBUILD, or by use the drag-and-drop to drag the cutline directly from the MASKVIEWS previewer. At runtime, you can load a cutline file from the “go simulator” line.

To load a cutline file using the popup:

1. Create and save a cutline file from MASKVIEWS. See Chapter 10: “MaskViews” for more information on how to do this. Typically, the first time through MASKVIEWS would be started from DECKBUILD, create or load a layout, then interactively create a cutline and save it to a file. For later use, you can go straight to step 2.
2. Bring up the **MaskViews Cut Files** popup (Figure 4-23) and set **Category** to **Disk Files**, the default. Choose the cutline file name in the scrolling list. If the file name does not appear, you may need to change the directory and filter on the popup. Click on **Refresh** to refresh the contents of the scrolling list if a new file were just created. You can also enter the name of the file next to **Filename** and click on **Load**.

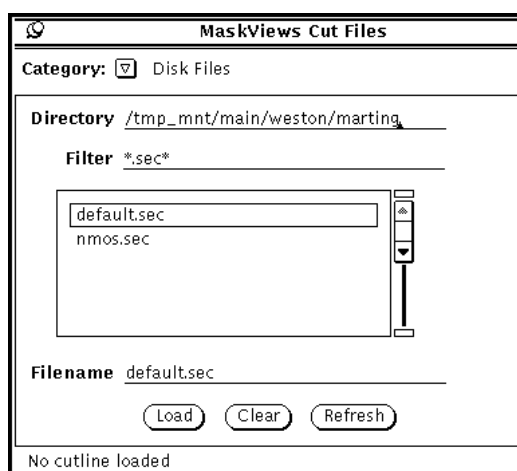


Figure 4-23: MaskViews Cuts Files - Disk Files Category

To load a cutline file via drag-and-drop:

1. Create a cutline file from MASKVIEWS. After either writing or previewing the mask, the cutline masks will be shown on the **2D masks** cutline viewer popup.
2. Bring up the **MaskViews Cut Files** popup (Figure 4-24) in DECKBUILD and set **Category** to **Drag & Drop**. Both this popup and the cutline viewer popup from MASKVIEWS must be visible on your screen.
3. Click and hold the **SELECT** menu button anywhere over the colored masks on the cutline viewer popup in MASKVIEWS. Still holding down **SELECT**, drag the mouse cursor into the large white area in the **Cut Files** popup in DECKBUILD. While dragging, the mouse cursor changes into a special cutline cursor to confirm the process of dragging a cutline. With the cursor over the **Cut Files** popup, release **SELECT**. The cutline information “drops” onto the popup.
4. To load the dropped cutline, click **SELECT** once on the **cutline** icon and click on **Load**. Selected icons are shown surrounded by a square box.

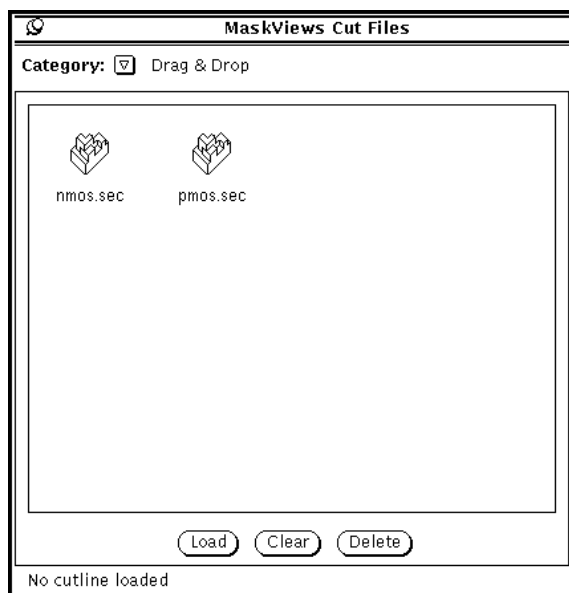


Figure 4-24: MaskViews Cut Files - Drag and Drop Category

To load a cutline from the input deck:

1. Create and save a MASKVIEWS file as specified in step 1 for loading a cutline disk file using the popup.
2. Use the syntax “cutline=filename” in the “go simulator” line to load the previously saved file. The following line loads a cutline stored in the file /default.sec and starts ATHENA.

```
go athena cutline="/usr/jdoe/default.sec"
```

It is possible to drag and drop up to 16 different cutlines this way (that’s all there is room for in the icon drop area). You do not need to save the cutlines to a file to use drag-and-drop. But if the same cutline is to be used again in the future, then it needs to be saved. Save cutlines from MASKVIEWS by clicking on **Write** on its popup (Figure 4-25). Either method of loading a cutline loads the mask information into DECKBUILD, and causes the mask names to appear in the SSUPREM3 and ATHENA **Mask** popups (Figure 4-28).

Note: You can clear the currently loaded cutline by either selecting the clear button in the MaskViews Cut Files popup or by using the “cutline=none” syntax in the “go simulator” line.

4.8.3: Starting Text Editor

Use the **Text Editor...** choice to startup the general **Text Editor** application with the file currently highlighted within the deck. The Editor is only invoked if a valid filename is provided. Otherwise, an error is displayed.

4.8.4: Starting Manager

Use the **Manager...** choice to start up the VWF INTERACTIVE TOOLS MANAGER. The menu item is placed here as a convenience feature.

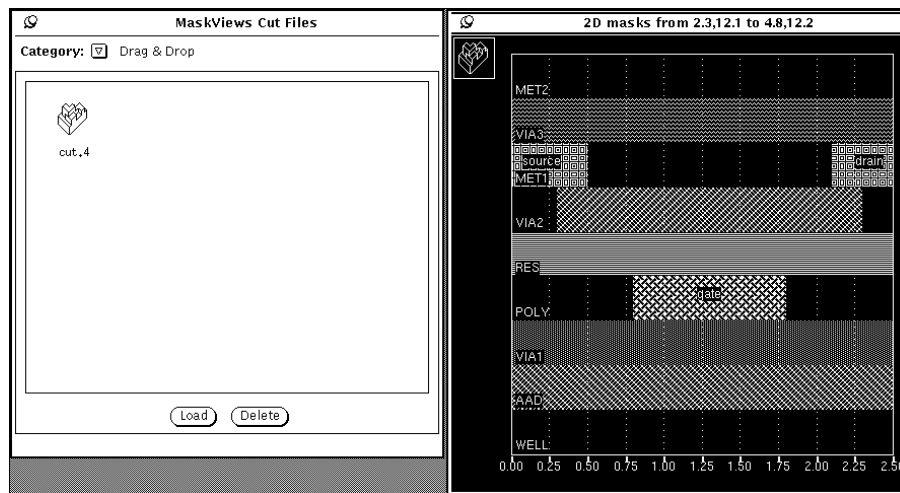


Figure 4-25: Drag and Drop from MaskViews

4.9: History

4.9.1: Overview

The **History** function allows moving backwards to any previous line in the input deck and restart execution. It is especially useful when debugging new decks, performing “what if” simulations, and in visualizing the device at different stages in the process flow.

DECKBUILD maintains a set of history files saved from the simulator as the simulation progresses. This permits going back to any previous step in the process by simply clicking on a line in the input deck. DECKBUILD automatically re-initializes the simulator with the correct history file.

4.9.2: History Control

You can configure how and if DECKBUILD maintains history files with the **History** popup. Click the **History Props...** button from the **Control Pad** category on the **Main Control** popup and the History popup appears (see Figure 4-26).

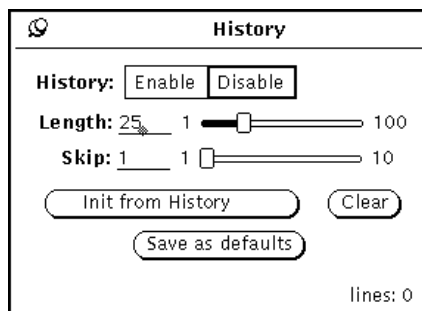


Figure 4-26: The History Popup

The **History** setting turns the entire mechanism on or off. If enabled, DECKBUILD saves history files after each significant simulation step. If disabled, history files are not saved and the simulation runs a little faster. History can be turned on and off as the simulation progresses. DECKBUILD allows re-initializing from any part of the deck that was run while history was enabled. History is enabled by default.

Length determines how many trailing process flow steps to remember (how many history files to maintain). The default is 25, although up to 100 steps may be saved. When the limit is reached, DECKBUILD starts re-using the old history files in a loop.

Skip determines how often history files are saved. The default value of 1 indicates that every significant process step causes a history file to be saved. A value of 2 indicates every other step, and so on. Significant process steps are implant, diffuse, etch, deposit, initialize/load, profile, and certain other statements. Comments, plot statements, blank lines, and certain other details are ignored. History files are not saved during device simulation.

Path, if activated, specifies the directory that the saved history files are saved in. This may be useful if large structures are being simulated and disk space for the current working directory is not abundant.

Note: If a simulation is executed with history and the path then altered, history initialization for the previous simulation fails.

Compress is switched off, by default, but when on all history files are compressed and appears in the form `history%.str.gz`. These files are automatically decompressed for initialization from history or loading into TONYPLOT. This function may be useful when simulating large or complex structures.

Clicking **Save as defaults** saves the current settings for use the next time DECKBUILD is run.

Initializing From History

After running part, or all the way through the deck with history enabled, the simulator can be re-initialized in the state it was in at some previous point in the deck. Re-initialize by selecting (highlighting) the line of interest, then clicking on the **Init** button on the Execution Control panel on the main window between the text and tty subwindows. For backwards compatibility, the **Init from History** button on the Main Control popup also provides this feature.

Note: This overloads the functionality of the Init button: if you select a filename which exists in the current directory instead, DeckBuild causes the simulator to load that file).

DECKBUILD also resets the current line to the selected line.

DECKBUILD may not have any history attached to the selected line if a comment line has been selected or skipped. DECKBUILD displays a notice prompt and suggest a previous point in the input deck by highlighting it. If selecting a line that is so far back in the deck that DECKBUILD no longer maintains relevant history, a notice prompt appears to inform you of the condition. Use a line closer to the current position or increase the history length.

When re-initializing from history, any “go simulator” flags (see the Sections 4.10: “Auto Interfacing” and 4.16.7: “GO”) specified on the go statement associated with the selected line are also re-initialized. For example, if a MASKVIEWS cutline file had previously been loaded using the syntax cutline=filename, then the specified file would be reloaded into DECKBUILD.

Removing History Files

Since history files can take up a fair amount of storage space, DECKBUILD provides two ways to remove them. First is to delete them at any time during the run by clicking on **Clear** in the **History** popup. Second, let them be removed when DECKBUILD is quit.

DECKBUILD is configured by default to remove history files at quit time, and displays a notice prompt to confirm their deletion. Change the default by changing the **Remove history files** setting on the **Options** category of the **Main Control** popup. The choices are to remove history files, confirm their deletion, or not to remove them at all.

In any case, history files are always saved in and removed from the current directory.

4.10: Auto Interfacing

4.10.1: Overview

Auto interfacing is the term used to describe DECKBUILD's capability of automatically transferring simulation data between different simulators. Simulation of a device may proceed transparently from SSUPREM3 (1D process), through ATHENA (2D process), and finally to ATLAS (2D device). The thread of control can be transferred to any simulator under DECKBUILD, including DEVEDIT for interactive mesh adaptation. Auto Interfacing enhances the power of simulation by allowing concentration on which simulator is best for the job, rather than on how to get one simulator to talk to another. Silvaco standards of simulator commonality is based on products that use a common data format.

4.10.2: Scenario

A typical simulation flow is shown in Figure 4-27.

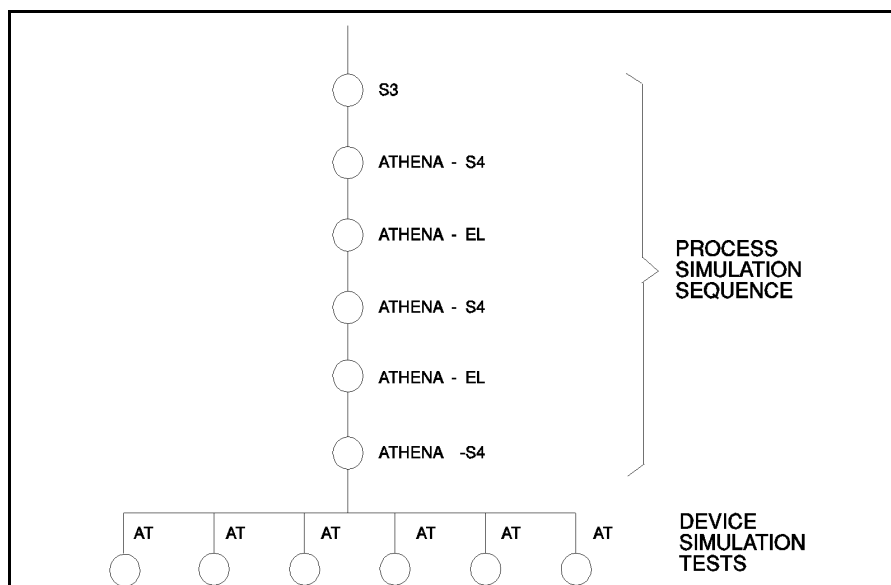


Figure 4-27: The History Popup

The 2D simulation starts with a 1D process simulator (SSUPREM3), since the initial processing of the device is entirely planar. For example, the initial well processing of a MOS device can be considered in 1D until the gate poly is etched. At the point when 2D is first required, an auto interface statement is placed in the deck, followed by 2D mesh definition and mesh initialization commands. At run time, DECKBUILD automatically transfers the 1D data from SSUPREM3 onto the 2D ATHENA mesh.

Alternately, the auto mode of ATHENA can be used. This mode automatically performs a 1D simulation in ATHENA until any statement requiring 2D is encountered (such as an etch left/right). ATHENA then transfers automatically into 2D mode for the remainder of the simulation. You benefit from applying the same models and syntax throughout the process simulation. The choice of SSUPREM3 or 1D ATHENA depends on which simulator is preferred, or for which special models have developed.

Once in ATHENA, you can continue process simulation and interface to other 2D simulators as well. It is also possible to interface to DEVEDIT at any time to adaptively remesh the device in preparation for device simulation.

Finally, a number of ATLAS device tests are shown at the end of the process sequence. Unlike the process sequence, where each section acts as a link in the chain of processing, the device tests each act on the final process structure. Append as many device tests as needed to the end of a process simulation and each will use the same final process structure as input.

The Active Structure

This final structure at the end of process simulation is called the active structure. DECKBUILD saves and remembers the active structure whenever auto interfacing is performed from a process simulator to any other simulator. Device tests always use the active structure unless explicitly initialized otherwise.

The current active structure is always shown on the left footer of the Main Control popup.

The Auto Interface Statement

The place in the input deck where auto interfacing should occur is marked by inserting an auto interface statement. The statement looks like

```
go simulator
```

where simulator is any valid simulator name. Consider the following input deck fragment that interfaces from SSUPREM3 to ATHENA:

```
GO SSUPREM3
#
INIT SILICON THICK=1.2 SPACE=500 BORON CONC=1E14
#
DIFFUSE TEMP=100 TIME=20 WETO2
#
IMPLANT PHOSPHORUS DOSE=1E13 ENERGY=40 PEARSON
#
GO ATHENA
#
LINE Y LOC=0.0 SPAC=0.2 TAG=TOP
LINE Y LOC=0.50 SPAC=0.10
LINE Y LOC=1.00 SPAC=0.15 TAG=BOTTOM
#
LINE X LOC=0.00 SPAC=0.10 TAG=LEFT
LINE X LOC=1.00 SPAC=0.10 TAG=RIGHT
#
INIT ORIENTATION=100 AUTO
#
ETCH OXIDE RIGHT P1.X=0.2
```

Here, an oxide in 1D is grown using SSUPREM3 and transfers control to ATHENA to perform the 2D etch. At run time, the 1D doping profile is automatically transferred from SSUPREM3 onto the 2D ATHENA mesh, and oxide deposited on top. The oxide profile is transferred.

Probably the best way to create an auto interface statement is to have DECKBUILD create it automatically. This is done by placing the text caret in the text subwindow at the point desired to insert the statement. Then, enable the **Write to Deck** choice on the **Control Pad** and click on **Select Current Simulator** to write the auto interface statement. Usually, the most convenient time to do this is when finished writing statements for one simulator and beginning to write statements for the

next. DECKBUILD does not only insert the auto interface statements but also brings up the proper **Commands** menu for the new simulator.

How Auto Interface Works

When DECKBUILD gets a request to perform an auto interface (from an `auto interface` statement in the input deck or through the **Control Pad**), it evaluates whether an interface is appropriate: 1D to 2D process is legal, but 2D to 1D process is not. If an interface is appropriate, then DECKBUILD also checks to see if the current simulator has been initialized or not. For example, if ATHENA has not yet executed an `initialize` statement, it doesn't have any simulation results to pass on to the next simulator. Finally, if both these conditions are satisfied, then DECKBUILD causes the current simulator to save its simulation data, shut down the current simulator, starts up the new simulator, and initializes the new simulator with the saved data. If either condition is not satisfied, then DECKBUILD honors the request to start up the new simulator, but does not attempt to initialize it with saved simulation data. The latter is appropriate when moving backwards in an input deck, and for quick "look and see" experiments with another simulator.

You can also alter the default input and output flags using the `go simulator` interface statement. For example, the default `load` statement for DEVEDIT includes the **mesh** flag. Using the following syntax, the mesh can be loaded on auto interface:

```
go devedit inflags = "mesh"
```

Simulator flags can be appended to the existing default flags and MASKVIEWS cutlines can also be loaded automatically in the deck using the **simflags** and **cutline** arguments respectively. For more information and examples, see Section 4.16.7: "GO".

4.11: IC Layout Interface

4.11.1: Overview

The **IC Layout Interface** (MASKVIEWS) allows the building of a deck that can be used to run a cross-section from any region on the layout. Such a deck is called a generic deck. A generic deck is always used in conjunction with the **IC Layout Interface**, which consists of loading cutline information into DECKBUILD. The cutline information contains location-specific masking information from a 1D or 2D cutline across the surface of the layout, taken from MASKVIEWS.

To use the **IC Layout Interface**, create a layout and a corresponding generic deck. The generic deck uses mask information defined on the layout and is identical in nature to the “run sheet” used in the fab. Unlike the typical process simulation input deck, it defines the order of process steps and interweaves the mask steps.

When the generic deck is complete, choose a one or two-dimensional cross-section in MASKVIEWS over the layout. This cross-section is known as a cutline. Load the cutline into DECKBUILD and click on the **Run** button. DECKBUILD automatically uses the cutline information to substitute mask, mesh, region, and electrode information at run time.

For information on how to start MASKVIEWS and load cutlines, see Section 4.8: “Tools”. For information on how to create a layout, see Chapter 10: “MaskViews”.

4.11.2: Creating a Generic Deck

Generic decks have no geometry information, use masks, optionally use regions and electrodes, and have no horizontal grid information for 2D mesh generation. All of this information is quite specific to the cross-section that has been chosen through the layout and is contained in the cutline file. The horizontal grid information is calculated by MASKVIEWS so that a line with a user-specified grid spacing is placed at each mask edge and is substituted by DECKBUILD at run time.

Mask Statements

The best place to start writing a generic deck is with mask definitions. The first step is to initialize the Mask popup with the correct mask names to write the deck properly. To do this, invoke MASKVIEWS from DECKBUILD using the **Tools** menu, then create or load the layout. Create a cross-section and store it to a cutline file, then load that file into DECKBUILD as shown in Section 4.8: “Tools”.

At this point, the **Mask** popup (Figure 4-28) should display all the masks in a scrolling list. Pinning the popup keeps it from disappearing each time a mask is written to the deck.

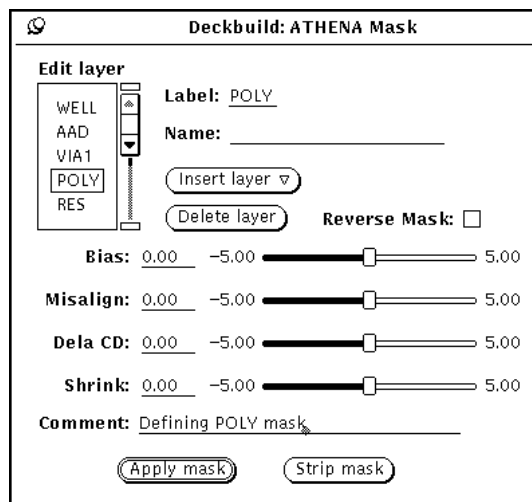


Figure 4-28: The Mask Popup

Write the deck as much as it the process flow appears in the fab. Use masks to structure the deck so that it is capable of producing any device on the layout. Insert mask statements by selecting the desired mask on the scrolling list, then clicking on **WRITE**.

At runtime, the mask statements are substituted with either `deposit` or `etch` or both statements or **Optolith** layout statements when the `optolith` flag is used with the `mask` command

MASKVIEWS allows the definition of either barrier or photoresist as the masking material. Barrier is a fictitious material that behaves like a perfect photoresist and can be deposited in very thin layers (about 0.02 microns) to save grid points. Barrier is recommended in all situations except when photoresist penetration studies are required.

4.11.3: Regions

You can also define regions in MASKVIEWS. Regions are boolean combinations of masks that uniquely define an area on the layout. Regions are used in `extract` statements in place of `x.val` to define the x-location where the desired quantity is to be extracted. For example, a layout for a MOS inverter that contains n- and p-type devices may have WELL, AAD, and POLY masks. To measure the gate oxide thickness in both types of devices, one approach might be to define a region GATE in MASKVIEWS where WELL is “don’t care”, AAD is “true”, and POLY is “true”. All other masks would be “don’t care”. Then, use this region in an `extract` statement:

```
extract oxide thickness region="GATE"
```

rather than:

```
extract oxide thickness x.val=1.0
```

Electrodes

Electrode positioning is the last remaining area of the deck that requires layout-specific values. In non-generic decks, an `electrode` statement in ATHENA requires both a specific x-location value and a name for that electrode. For example:

```
electrode name="gate" x.val=1.0
```

A generic deck can obviously assume neither the x-location of an electrode, its name, nor even how many electrodes there are. The `autoelectrode` statement provides the solution to automatically place and name electrodes.

Use MASKVIEWS to define which masks are electrode masks and the corresponding electrode names. Electrode masks are those masks that are used for defining one or more electrodes. For instance, the POLY mask for a MOS device is an electrode mask because it forms the gate contact. Along with the electrode attribute, MASKVIEWS allows the specifying of an electrode name for a mask (or part of a mask) as well. Electrode names and masks should be specified in MASKVIEWS before generating a cutline file in MASKVIEWS.

To use the electrode masks, enter the `autoelectrode` statement in the input deck directly after contact definition using the mask of interest. `Autoelectrode` takes no parameters or arguments. It works by inserting `electrode` statement(s) at run time using information corresponding to the last electrode mask. If the same mask is used to define more than one contact, use MASKVIEWS to assign a separate name for each section of the mask. DECKBUILD substitutes a separate `electrode` statement for each contact. Thus, a single `autoelectrode` can generate multiple `electrode` statements.

Figure 4-29 shows the use of `autoelectrode` in a generic deck. Notice that DECKBUILD automatically comments out the mask and `autoelectrode` statements from the deck as they are executed by the simulator. The comments only appear in the run-time output; the deck itself is not changed.

Note: DECKBUILD only remembers the electrodes specified within each mask. Therefore, an `autoelectrode` statement must be used for every mask layer where electrodes are defined. This defines multiple electrodes for a single `autoelectrode` statement within the current mask. In other words, both the source and drain of a MOS transistor could be located on the same metal level.



Figure 4-29: DeckBuild Main Window

If structure files are saved after masking and electrode steps, they must be saved after the `autoelectrode` statement, and not in between the mask statement and the `autoelectrode` statement. Otherwise, the structure file does not contain the electrode information (which is inserted by the `autoelectrode` statement). In this case, the only way to add it would be to go through the masking operation a second time as shown in the following example.

```

mask name="POLY"

etch poly dry thick=0.4

strip

autoelectrode

struct outf="poly.str"
  
```

Do not do this:

```

mask name="POLY"

etch poly dry thick=0.4

strip

struct outf="poly.str"
  
```

```
autoelectrode
```

Enabling

Enable cutline substitution by setting **Auto Maskviews** to **ON** on the **Main Control Options** popup. Substitution begins as soon as a cutline file is loaded from the **MaskViews Cut Files** popup. This popup can be accessed from the **Tools** menu. See Section 4.8: “Tools” for information on loading a cutline.

Disable substitution by turning **Auto MaskViews** to **OFF**. No substitution occurs even if a cutline file is loaded.

Cutlines can also be loaded and cleared from DECKBUILD at runtime using the go simulator `cutline=filename` syntax. DECKBUILD loads the MASKVIEWS cutline from the current working directory if you do not specify path. DECKBUILD clears an existing cutline if you specify `cutline=none`.

4.11.4: Rules of Thumb

To make sure that your deck is indeed generic, obey the following rules:

1. etch statements are either

```
etch dry material thickness value
```

or

```
etch material all
```

2. Do not use

```
etch left material pl.x=value
```

3. Do not include horizontal mesh information for 2D process simulators. The mesh information is substituted automatically at run time.

4. Use mask statements where photoresist is required:

```
mask name="PWELL"
```

or

```
mask name="AAD" reverse
```

5. Beware of wedges. As in real experiments, wedges, fillets, and spacers may appear if anisotropic (dry) etches are not used carefully.

6. Use region in extract statements rather than x values.

```
extract oxide thickness region="GATE"
```

7. Use autoelectrode statements rather than electrode statement

4.11.5: Mask Bias, Misalignment, and Delta CD

Mask, bias, misalignment, and delta CD information can be entered in the mask statement to study these effects, either standalone or as experimental variables in the VWF environment.

Bias, misalignment, and delta CD can be studied together or independently. Misalignment shifts the entire specified mask left (negative numbers) or right. Bias and delta cd increase or decrease the width of the mask (for positive masks, a positive bias/delta cd will decrease the width of the etched hole(s) in the mask).

A value of 0 for any parameter is equivalent to not specifying that parameter. The values have units of microns.

The syntax is:

```
mask name="<region_name>" [bias=<value>]/
```

```
[misalign=<value>] [delta_cd=<value>]
```

4.11.6: Using DevEdit with IC Layout

DEVEDIT can use mask region information when running under DECKBUILD to place mesh constraints in areas specified by MASKVIEWS regions. An example would be to place a finer mesh in the region under the gate of a MOS device. This can be specified from the **Mesh Constraints** popup of DECKBUILD. The resulting syntax is:

```
constrain.mesh under.mask="regionname" other_parameters...
```

Note: `under.mask` is a misnomer. The specified area is a MASKVIEWS region, which is a boolean combination of perhaps several masks, so the region may not be wholly under or composed of a single mask. It is really under a region, not a mask. To eliminate confusion with DEVEDIT material regions, use the parameter `under.mask`.

4.12: UTMOST Interface

4.12.1: Overview

The UTMOST interface allows Silvaco's parameter extraction package UTMOST III to load in data from one of more device simulation runs and perform SPICE model parameter extraction. With the VWF, in particular, this allows you to generate response surfaces that model SPICE parameters as a function of process variations, including study of failure analysis, process synthesis, yield analysis, and with the SPAYN interface, worst case modeling.

4.12.2: Setting Up An UTMOST Input Deck

Although UTMOST normally runs as an interactive X-based program, you can run without its graphical interface and read commands from an interactive prompt when run under DECKBUILD like the other simulation tools.

UTMOST Input Deck Example

```
##### START UTMOST SIMULATION #####

go utmost

utmost type = mos

#
load Utmost model filemodel bsim3_pmos
#
set value for TOX
set tox_in_m = $tox*1.0e-10
device TOX = $tox_in_m
#
define device specifications
setup width = 1.0 length = 0.6 polarity = P
#
load Atlas log files
init inf= IdVg-Vb.log master
init inf= IdVd-Vg.log master append
#
select required characteristics for device
deselect ID/VG-VB all
select ID/VG-VB device 1
deselect ID/VD-VG all
select ID/VD-VG device 1
log outf = mos.log utmost
#
perform simulation
fit ID/VG-VB
```

```
fit ID/VD-VG

##### EXTRACT UTMOST PARAMETERS #####

extract name="P-VTH0" param="VTH0"
extract name="P-K1" param="K1"
extract name="P-K2" param="K2"
extract name="P-K3" param="K3"
extract name="P-W0" param="W0"
extract name="P-NLX" param="NLX"
extract name="P-DVT0" param="DVT0"
extract name="P-DVT1" param="DVT1"
extract name="P-UA" param="UA"
extract name="P-UB" param="UB"
extract name="P-UC" param="UC"
extract name="P-VSAT" param="VSAT"
extract name="P-A0" param="A0"
extract name="P-A1" param="A1"
extract name="P-A2" param="A2"
extract name="P-RDSW" param="RDSW"
extract name="P-VOFF" param="VOFF"
extract name="P-NFACTOR" param="NFACTOR"
extract name="P-PCLM" param="PCLM"
extract name="P-PDIBL1" param="PDIBL1"
extract name="P-PDIBL2" param="PDIBL2"
extract name="P-DROUT" param="DROUT"
extract name="P-PSCBE1" param="PSCBE1"
extract name="P-PSCBE2" param="PSCBE2"
extract name="P-TOX" param="TOX"
extract name="P-XJ" param="XJ"
extract name="P-U0" param="U0"
extract name="P-ETA" param="ETA"
extract name="P-Ilinm" param="Ilinm"
extract name="P-Ilins" param="Ilins"
extract name="P-Isatm" param="Isatm"
extract name="P-Isats" param="Isats"

quit
```

There are several points that must be followed to set up an input deck.

The first non-comment statement after the `go utmost` command must be `UTMOST TYPE=type` where `type` is either MOS, BIP, DIODE, GAAS, or JFET.

When the `UTMOST` statement is encountered, the correct `UTMOST` module is executed. If no `UTMOST` statement is encountered, the MOS module is run by default.

Note: The `UTMOST` command (`utmost type=type`) can be replaced by specifying the module type command line flag in the `go utmost` as below. This will append the `"-bip"` flag to the default `UTMOST` argument and start the correct module immediately. For example, `go utmost simflags="-bip"`.

The next non-comment statement must be a `model` statement. The `model` statement gives the model file name that `UTMOST` reads. The model file is assumed to exist in the `$SILVACO/var/utmost` directory. Therefore, the statement

```
MODEL BSIM3_P MOS
```

reads in the file `$SILVACO/var/utmost/bsim3_pmos`.

To create this model file, run a baseline device simulation, load its results (IV curves) into `UTMOST` interactively, perform the necessary modeling to generate the desired parameters, then save the model file from `UTMOST`. Once this has been done with a baseline device, the same model file can be used for similar devices. For example, a large process variation experiment on a device in the VWF can use the same model file for testing all variations.

Note: You need different model files for different devices. In other words, *n* versus *p* MOSFETs.

Use the `INIT` command to explicitly load one or more simulated IV result files:

```
INIT INF=filename MASTER append
```

The master flag tells `UTMOST` that the data is in master, or SSF, file format (the default output format from ATLAS). Use the append flag for loading all but the first IV file, as shown above.

Note: Load the files explicitly. The **auto-interface** feature of `DECKBUILD` is not active in `UTMOST` mode, and does not load them automatically.

Place `extract` statements at the end of the deck to extract the modeled SPICE parameters. The format of the `extract` statement is:

```
extract name="name" param="utmost_parameter_name"
```

where `name` is any name of your choice, and `utmost_parameter_name` is the name of an `UTMOST` parameter.

It is helpful to first run the deck without any `extract` statements, but with a `save outfile=filename` command. `UTMOST` always prints a list of all parameters it has modeled before it saves the file. You can then copy and paste directly from this list into your `extract` statements. The parameter names are case sensitive and the `extract` parameter name should match the `UTMOST` parameter name exactly.

Another noteworthy point in the example concerns the device TOX setting, which is taken as the value of \$tox multiplied by a conversion factor. \$tox is an extracted value taken from a gate oxide thickness measured in the ATHENA process simulation in the same input deck. \$tox is also used in a set statement to convert its units from angstroms (always used by extract to measure material thicknesses) to meters (used by UTMOST). The set statement performs the arithmetic, not in a simulator statement. Finally, the value of the result is substituted in the device TOX command.

In the case of the VWF, the \$tox thickness is measured in an input deck fragment run separately from the UTMOST input deck. The VWF remembers the value of \$tox (and all other extracted variables) and passes it down to all other children fragments. Therefore, it can still be used at any later point in the simulation thread.

In the VWF environment, the init infile filenames are substituted with automatically-generated filenames that contain all IV simulation data from all device simulations that feed into the UTMOST test. See the VWF manual for more information on how to connect device tests to UTMOST tests. It is important to note that the VWF substitutes all I-V data sets saved from all connected device tests (all data that was saved with a 'log outfile' command in the device tests). Unusual results may occur in the case where many extraneous files have been saved in the device tests. In that case, remove the extraneous log outfile statements as necessary.

Runtime Output Example

```
UTMOST> ##### START UTMOST SIMULATION #####
UTMOST> utmost type = mos
U T M O S T I I I
P A R A M E T E R E X T R A C T I O N S O F T W A R E
Version: 10.04 (Batch-mode)
Preliminary Version
Copyright 1989, 1990, 1991, 1992, 1993, 1994
SILVACO International
All rights reserved
=====
MOS Module: enabled
BIP Module: disabled
JFET Module: disabled
Diode Module: disabled
GAAS Module: disabled
Fitting Routines : enabled
Local Optimization : enabled
Global Optimization : enabled
Simulation : enabled
=====
Fri Jul 29 16:25:36 1994
Executing on host: elvis
UTMOST>
```

```
UTMOST># load UTMOST model file
UTMOST>model bsim3_pmos
  SETUP FILE LOADED. Version number: 66
UTMOST>
UTMOST># set value for TOX
UTMOST>## set tox_in_m = $tox*1.0e-10
UTMOST>device TOX = 1.79e-08
UTMOST>
UTMOST># define device specifications
UTMOST>setup width = 1.0 length = 0.6 polarity = P
UTMOST>
MOST># load Atlas log files
UTMOST>init inf= IdVg-Vb.log master
UTMOST>init inf= IdVd-Vg.log master append
UTMOST>
Log file loaded
UTMOST># select required characteristics for device
UTMOST>deselect ID/VG-VB all
UTMOST>select ID/VG-VB device 1
UTMOST>deselect ID/VD-VG all
UTMOST>select ID/VD-VG device 1
UTMOST>UTMOST>log outf = mos.log utmost
UTMOST>
UTMOST># perform simulation
UTMOST>fit ID/VG-VB
  Please wait, FITTING in progress!
  Executing local optimization ivgs_bsim3_a
  Executing local optimization ivgs_bsim3_a
UTMOST>fit ID/VD-VG
  Please wait, FITTING in progress
  Executing local optimization ivds_bsim3_a
  Executing local optimization ivds_bsim3_a
UTMOST>
UTMOST># output UTMOST parameters for extraction (used only for setup)
UTMOST>#save outf = mos.ssf
UTMOST>
UTMOST>##### Extract UTMOST parameters #####
```

UTMOST>

UTMOST>save outfile=/tmp/deckbEAAa06379

The following parameters have been stored:

VTH0	= -0.6363853	[V]	K1	= 0.9019918	[V^0.5]
K2	= -0.0688711	[.]	K3	= 65.72977	[.]
W0	= 4.883524E-6	[m]	NLX	= 4.674296E-8	[m]
DVT0	= 3	[.]	DVT1	= 0.452118	[.]
UA	= 6.048951E-10	[m/V]	UB	= 1E-22	[(m/V)^2]
UC	= -0.0253118	[1/V]	VSAT	= 5.8867E6	[cm/sec]
A0	= 0.5285819	[.]	A1	= 0.0539283	[1/V]
A2	= 0.6716171	[.]	RDSW	= 800	[ohm*m^3]
OFF	= -0.0139744	[V]	NFACTOR	= 1.3974359	[.]
PCLM	= 5.7002197	[.]	PDIBL1	= 0.0545063	[.]
PDIBL2	= 0.0610933	[.]	DROUT	= 0.1460536	[.]
PSCBE1	= 9.97005E9	[V/m]	PSCBE2	= 1E-9	[V/m]
TOX	= 1.79E-8	[m]	XJ	= 1.5E-7	[m]
U0	= 213.4599111	[cm^2/V/sec]	ETA	= 0.3	[.]
Ilinm	= 2.009946E-5	[A]	Ilins	= 1.99748E-5	[A]
Isatm	= 2.773636E-4	[A]	Isats	= 2.76221E-4	[A]

UTMOST>

EXTRACT> init infile="/tmp/deckbEAAa06379"

EXTRACT> extract name="P-VTH0" param="VTH0"

P-VTH0=-0.636385 V

EXTRACT> extract name="P-K1" param="K1"

P-K1=0.901992 V^0.5

EXTRACT> extract name="P-K2" param="K2"

P-K2=-0.0688711

EXTRACT> extract name="P-K3" param="K3"

P-K3=65.7298EXTRACT> extract name="P-W0" param="W0"

P-W0=4.88352e-06 m

EXTRACT> extract name="P-NLX" param="NLX"

P-NLX=4.6743e-08 m

EXTRACT> extract name="P-DVT0" param="DVT0"

P-DVT0=3

EXTRACT> extract name="P-DVT1" param="DVT1"

P-DVT1=0.452118

EXTRACT> extract name="P-UA" param="UA"

```
P-UA=6.04895e-10 m/V
EXTRACT> extract name="P-UB" param="UB"
P-UB=1e-22 (m/V)^2
EXTRACT> extract name="P-UC" param="UC"
P-UC=-0.0253118 1/V
EXTRACT> extract name="P-VSAT" param="VSAT"
P-VSAT=5.8867e+06 cm/sec
EXTRACT> extract name="P-A0" param="A0"
P-A0=0.528582
EXTRACT> extract name="P-A1" param="A1"
P-A1=0.0539283 1/V
EXTRACT> extract name="P-A2" param="A2"
P-A2=0.671617
EXTRACT> extract name="P-RDSW" param="RDSW"
P-RDSW=800 ohm*m^3
EXTRACT> extract name="P-VOFF" param="VOFF"
P-VOFF=-0.0139744 V
EXTRACT> extract name="P-NFACTOR" param="NFACTOR"
P-NFACTOR=1.39744
EXTRACT> extract name="P-PCLM" param="PCLM"
P-PCLM=5.70022
EXTRACT> extract name="P-PDIBL1" param="PDIBL1"
P-PDIBL1=0.0545063
EXTRACT> extract name="P-PDIBL2" param="PDIBL2"
P-PDIBL2=0.0610933
EXTRACT> extract name="P-DROUT" param="DROUT"
P-DROUT=0.146054
EXTRACT> extract name="P-PSCBE1" param="PSCBE1"
P-PSCBE1=9.97005e+09 V/m
EXTRACT> extract name="P-PSCBE2" param="PSCBE2"
P-PSCBE2=1e-09 V/m
EXTRACT> extract name="P-TOX" param="TOX"
P-TOX=1.79e-08 m
EXTRACT> extract name="P-XJ" param="XJ"
P-XJ=1.5e-07 m
EXTRACT> extract name="P-U0" param="U0"
P-U0=213.46 cm^2/V/sec
```

```
EXTRACT> extract name="P-ETA" param="ETA"  
P-ETA=0.3  
EXTRACT> extract name="P-Ilinm" param="Ilinm"  
P-Ilinm=2.00995e-05 A  
EXTRACT> extract name="P-Ilins" param="Ilins"  
P-Ilins=1.99748e-05 A  
EXTRACT> extract name="P-Isatm" param="Isatm"  
P-Isatm=0.000277364 A  
EXTRACT> extract name="P-Isats" param="Isats"  
P-Isats=0.000276221 A  
EXTRACT> quit  
UTMOST>quit  
UTMOST finished  
*** END ***
```

4.13: SmartSpice Interface

The SMARTSPICE interface allows SILVACO's circuit simulation program to execute within DECKBUILD and the VWF AUTOMATION TOOLS. The interface reads through the whole deck \$-substituting any variables that have been set. The actual simulation does not occur until the `solve outfile=<rawfile>` is reached in the simulation deck.

The output rawfile is a Data Format file that can be visualized in TONYPLOT or used with extract statements to obtain required measurements as shown below.

```
extract init infile="spice.dat"
extract name="curve1"
max (curve(da.value."vin", da.value."power"))
extract name="curve2"
max (curve(da.value."2" "vin", da.value."2" "power"))
```

These extract statements will return the maximum of power for the first and second data set in the file `spice.dat`.

4.14: Internal Interface

The **Internal** interface is provided as a split point area for **Device** simulation only experiments using VWF AUTOMATION TOOLS. This interface only accepts certain DECKBUILD statements (`set`, `tonyplot`, `extract`, `go`, `source`), the most useful is the `set` command. The **Internal** interface is intended to include `set` statements which define the required input values to be `$`-substituted into the device experiment. Using VWF AUTOMATION TOOLS, these inputs can be split on (varied) over many simulations to provide Device experiments without using process simulation.

4.15: Remote Simulation

4.15.1: Overview

DECKBUILD has the capability to be running on a local host while executing a simulation on a remote host. The simulation is run using a remote shell command while displaying the output back to the tty window in DECKBUILD.

In interactive mode, the remote host for each simulator can be specified using the Simulator Properties popup accessed from the Main Control popup. For batch mode (-run), you can use the -remote <hostname> command line option. This specifies the same remote host for all simulators used.

4.15.2: Remote Options

Within the **Main Control** popup under the Options category, there are some remote options that can be used to customize remote simulation.

Remote tmp directory sets the remote simulation tmp directory, which must be mounted on the host executing DECKBUILD and the host executing the simulator.

Remote shell command specifies the remote shell command to be used for remote simulation. This option may need to be set with a specific path such as:

```
Solaris2 & decalpha-osf1 /bin/rsh
Solaris1 /usr/ucb/rsh
rs6000-aix4 /usr/bin/rsh
hp700-hpux /usr/bin/remsh
mips64-irix6 /usr/bsd/rsh
```

Remote mount string removes the automount prefix (usually /tmp_mnt) from paths for remote simulation.

4.15.3: Troubleshooting

Remote simulation attempts to diagnose common problems when a simulator is started. Two remote shell routines are performed to check the following five items, if any are not correct the simulator is killed and an error message output to explain what is required.

- **Unknown host** - Check if remote host name is entered correctly in the **Simulator Properties** popup or on command line. If so, contact your System Administrator as you are unable to access the required remote host.
- **.rhost file Error** - For remote simulation, the name of the local host and your username must be entered into the .rhost file located in your home directory. To continue, either add the line <hostname> <username> to your .rhost file or contact your System Administrator to make the required changes.
- **CWD Mount Error** - For remote simulation, the current working directory must be mounted for the remote machine. To continue, either change the current working directory by loading your input file from a directory mounted for both machines or contact your System Administrator to ensure the present directory is mounted on the remote host.
- **Write Permission Error** - For remote simulation, write access must be available for the temporary directory. To continue, either change **Remote Tmp Directory** setting under the **Main Control** popup **Options** category to an accessible directory or contact your System Administrator to set the current remote tmp directory to be write accessible.
- **Tmp Dir Mount Error** - For remote simulation, the temporary directory must be mounted for both the local and remote machines. To continue, either change **Remote Tmp Directory** setting under the **Main Control** popup **Options** category to a directory mounted for both machines or contact

your System Administrator to ensure the current remote tmp directory is mounted on the remote host in addition to the local machine.

Note: Use of remote simulation is not recommended and not supported when `DECKBUILD` is executed from a remote machine and displayed locally. Either remote login to a machine and execute `DECKBUILD` and the simulators on that host or run `DECKBUILD` on your local machine and use remote simulators.

4.16: Statements

4.16.1: Overview

This section contains a complete description of every statement and parameter used by DECKBUILD. The following information is provided for each statement:

- The statement name
- The syntax of the statement with a list of all the parameters of the statement and their type
- A description of each parameter
- An example of the correct usage of each statement

4.16.2: DeckBuild Commands

The following list identifies the commands that DECKBUILD executes. Each of these commands is described in subsequent sections:

- ASSIGN
- AUTOELECTRODE
- DEFINE
- ELSE
- EXTRACT
- GO
- IF
- IF.END
- L.END
- L.MODIFY
- LOOP
- MASK
- MASKVIEWS
- SET
- SOURCE
- STMT
- SYSTEM
- TONYPLOT
- UNDEFINE

4.16.3: ASSIGN

Provides a much richer version of the functionality provided by the existing SET statement (see Section 4.16.12: “SET”).

Syntax

This is the syntax of the ASSIGN statement:

```
assign  name = <variable>           [print]

      (n.value = <expr_array> [delta=<expr> | ratio=<expr>] |
      l.value = <expr_array> |
      c.value = <qstring>           [delta=<expr>] |
```

```
<c_array>
)
```

```
[level = <expr>]
```

with the following subsidiary definitions :

```
<expr_array> -> <expr> |
                 (<expr>, <expr_array>) |
                 (<expr> <expr_array>)

<c_array>      -> c<integer>=<qstring> |
                 c<integer>=<qstring> <c_array>
```

Description

The ASSIGN statement allows you to assign either a numerical (n), a logical (l) or a character (c) value to a variable. Numerical values may be arbitrary arithmetical expressions and may incorporate any of the standard functions mentioned in Section 4.16.12: "SET". All user-defined variables will be substituted before the expression is evaluated.

Arbitrarily, many variables may be assigned in the same deck.

Logical values may also be arbitrary numerical expressions. If any expression evaluates to a non-zero value, it is interpreted as true. Otherwise, it is interpreted as false. You can use the actual words "true" and "false". You can also assign arbitrary boolean expressions to logical values. The following operators are recognized:

```
logical AND      &
```

```
logical OR      |
```

```
logical NOT      ^
```

The usual relational operators are also recognised (>, <, >=, <=) with a single '=' character for the equals operator and the token ^= for the not-equals operator.

Note: Although unquoted strings are supported, you should **always** use quoted strings for character values for the sake of clarity.

You can assign a whole array of values to a variable. Arrays of numerical and logical arrays are written in the following manner:

```
(1, 2, 4, 8)
```

but arrays of character variables are written like this :

```
c2 = "Mary" c3 = "had" c5 = "a" c7 = "little" c11 = "lamb."
```

You can have many terms in a character array with their defining integers (the ones prefixed with 'c' for 'character') and not be sequential.

The array will be sorted in the increasing order of its defining integers.

Arrays are usually assigned to variables in loops. After each loop, the next value in the array will be assigned to the variable. If the end of the array comes before the end of the loop, the variable will revert to the first value in the array on the next pass.

You can also use the delta and ratio clauses to alter a variable on each pass through a loop. If you specify delta, that value will be added to the variable on each pass. If you specify ratio, the variable will be multiplied by that value on each pass.

If you specify an array of values, you cannot then specify either the delta or the ratio clauses.

You can specify a delta clause for a character value. This increment must be an integer and will be truncated if it isn't. This is an odd concept but is useful when, for example, you want to use a new output file on each iteration of a loop. A few examples will illustrate the idea. If the character value is a00 and delta is 4, then the first few values the variable takes will be a00, a04, a08, a12 and so on. Eventually, you will reach the values a92, a96, b00, b04, and so on. Incrementing lower-case 'z' by one produces lower-case 'a' but not upper-case 'A' and vice versa. You can also specify a negative delta with the obvious results.

An ASSIGN will persist until you encounter a second ASSIGN with the same variable name. If this happens, the old ASSIGN will be discarded and replaced by the new one. If an ASSIGN is outside of all loops, then the value of its variable never changes. If it's inside a loop, then its variable changes every time a new iteration of the loop begins.

If you specify the print keyword, the current value of the variable will print when initialized and will change each time thereafter.

You can use the level clause to have the value of the variable change when a particular member of a set of nested loops begins a new iteration. If the level you specify is positive, the loop is obtained by counting downwards from the zero level, the one outside of all loops. If the level is negative, the loop is obtained by counting upwards from the current level towards the outermost loop. So, level=-2 means change when the loop two above the present one starts a new iteration. level=2 means change when the next-to-the-outermost loop begins a new iteration.

As already mentioned, user-defined variables will be substituted before attempting expression evaluation. These variables are defined using the SET and ASSIGN statements. You can indicate the presence of a user-defined variable by prefixing it with '\$' or '@' or by surrounding it with braces like this:

```
${my_variable_1}, @{my_variable_2}.
```

Variables embedded withing quoted strings will be correctly substituted. "Bare" variables will be recognized provided they are surrounded by both spaces and parentheses. This usage, however, is very confusing and highly inadvisable.

Examples

1. In this example, param1 will take the values 1, 2 and 3 on the three passes through the loop.

```
loop steps=3
  assign name=param1 print n.value = 1 delta = 1
l.end
```

2. This generates the sequence aa.20, aa.16, aa.12, aa.08, aa.04 and aa.00 for param2.

```
loop steps=6 print
  assign name=param2 c.value = "aa.20" delta = -4
l.end
```

3. Followed by, "Mary", "had", "a", "little" and "lamb".

```
loop steps=5 print
    assign name=param3 c10="lamb." c3="Mary" c8="little" c4="had" c7="a"
l.end
```

In the two preceding examples, the double quotation marks will **not** be included when param2 and param3 are substituted into later expressions.

4. param1 takes the values 42, 38, 17, 42, 38.

```
loop steps=5 print
    assign name=param1 n.value = (42, 38, 17)
l.end
```

5. param1 takes the values 42, 45.2, 48.4, 51.6, 54.8.

```
loop steps=5 print
    assign name=param1 n.value = 42 delta = 3.2
l.end
```

6. param1 takes the values 42, 134.4, 430.08, 1376.26, 4404.02.

```
loop steps=5 print
    assign name=param1 n.value = 42 ratio = 3.2
l.end
```

7. This is a simple example illustrating the use of boolean expressions.

```
assign name=condition l.value = ($x > 0.0 & $y < 3.0)
```

If *x* and *y* represent coordinates, the value of condition will be true or false accordingly as the coordinates are in a required area of the structure. The value of \$condition could then be used as input to an IF statement.

8. It is worth emphasizing that ASSIGN can be used for the simplest of cases. See the following example:

```
assign name=e_charge n.value=1.6e-19
```

4.16.4: AUTOELECTRODE

Defines layout-based electrodes

Syntax

```
autoelectrode
```

Description

The autoelectrode command causes DECKBUILD to submit electrode definition statements to the current simulator. The electrode name and positioning information will be taken from the MASKVIEWS layout data.

Note: DECKBUILD only remembers the electrodes specified within each mask. Therefore, an autoelectrode statement must be used for every mask layer where electrodes are defined. This defines multiple electrodes for a single autoelectrode statement within the current mask.

See

Section 4.11: “IC Layout Interface”.

4.16.5: DEFINE and UNDEFINE

DEFINE replaces all subsequent occurrences of an identifier with a specified string.

UNDEFINE cancels this action.

Syntax

```
define    <identifier> <rest_of_line>
undefine  <identifier>
```

Description

The identifier should either be a quoted string or a well-formed identifier. That is, one which begins with a letter or an underscore and continues with an arbitrary sequence of letters, digits, underscores and periods.

Every time this token is identified thereafter, it will be replaced by the whole of the rest of the DEFINE statement from the end of the token down to the end of the line. This <rest_of_line> component may consist of any characters whatsoever.

You don't have to flag the presence of the defined (DEFINE) token using a '\$' or '@' prefix or any of the other methods mentioned in Section 4.16.3: “ASSIGN”.

Substitution of a defined (DEFINE) token will persist until you encounter an UNDEFINE statement referencing the same token.

Substitution of defined (DEFINE) tokens will occur before each line is executed, unless the line begins with a % character. This also holds for the DEFINE and UNDEFINE lines themselves and has an odd corollary, which you can see in the examples section.

Examples

1. Here is a straightforward example:

```
define mypath /home/john_smith/tmp/logs
.
.
.
log outf=mypath/file1.log
.
.
.
log outf=mypath/file2.log
```

This pathology will define black as white.

```
define color black
.
.
.
define color white
```

To get the behavior you probably had in mind, do this :

```
define color black
.
.
.
%define color white
```

2. Something similar happens with the UNDEFINE command. In the next example, "black" is substituted for "color" in the UNDEFINE command and a no-op results.

```
define color black
.
.
.
undefine color
```

3. For an UNDEFINE to take effect, always use the '%' prefix. For example:

```
define color black
.
.
.
%undefine color
```

4.16.6: EXTRACT

Extracts information from the current simulation

Syntax

```
extract extract-parameters
```

Description

The `extract` statement is used to extract interesting information from the current simulation. See Chapter 5: “DeckBuild:Extract” for a complete description.

See

Chapter 5: “DeckBuild:Extract”.

4.16.7: GO

Interface between simulators

Syntax

```
go <simulator> [inflags=<> | outflags=<> | simflags=<> | cutline=<>|noauto]
```

Description

The `GO` statement tells `DECKBUILD` to shut down the current simulator and start up the specified simulator when the statement is executed. It is used to auto-interface between simulators.

simulator can be `ssuprem3`, `athena`, `sminimos4`, `atlas`, `devedit`, `utmost`, `neurofab`.

inflags specifies new load command flags for autointerface.

outflags specifies new save command flags for autointerface.

simflags specifies flags to be appended to default simulator argument.

cutline specifies a MASKVIEWS cutline file to be loaded into DECKBUILD.

noauto specifies that no autointerface occurs for this go statement.

Examples

If the current simulator is SSUPREM3, then this statement causes DECKBUILD to quit SSUPREM3 and start up ATHENA.

```
go athena
```

This will replace the default flags used in ATHENA auto interface command with "master" when loading and "flip.y" when saving.

```
go athena inflags=master outflags=flip.y
```

Note: One or more flags can be specified on the go line.

This statement will append "-V 2.2.1.R" to the default DEVEDIT argument to start version 2.2.1.R of the tool.

```
go devedit simflags="-V 2.2.1.R"
```

Note: Quotes are required where spaces used in flags or multiple flags used.

This loads the MASKVIEWS cutline default.sec from the specified directory into DECKBUILD.

```
go athena cutline="/usr/jdoe/default.sec"
```

This removes the currently loaded MASKVIEWS cutline.

```
go athena cutline=none
```

Note: The cutline flag should never be used with VWF.

The cutline flag cannot be used within VWF because is no guarantee that the specified directory path for the cutline file will exist on any of the remote machines in a network that VWF jobs can be sent to.

If the current simulator is ATHENA, then the following statement causes DECKBUILD to quit ATHENA and start up ATLAS but no autointerface between the two simulators will occur.

```
go atlas noauto
```

See

Section 4.10: "Auto Interfacing"

4.16.8: IF, ELSE and IF.END

These three commands together provide the standard IF block functionality.

Syntax

```
if cond = (<boolean_expr>)
else [if cond = (<boolean_expr>)]
if.end
```

Description

The IF command starts the block. If its condition evaluates to true, then statements down to the next ELSE or IF.END line will be executed. If the condition evaluates to false, then there will be a search for an ELSE IF line whose condition evaluates to true. If you find such a line, the lines in its sub-block will be executed. At most, one sub-block in a given IF block will be executed.

The <boolean_expression> can be an arbitrary combination of boolean variables concatenated with AND, OR or NOT operators as described in Section 4.16.3: "ASSIGN".

You can nest IF blocks with each other and with LOOPS. As usual, an ELSE or an IF.END is associated with the most recent IF. There is no mechanism for using brackets or braces to enforce a particular nesting.

Example

```
if cond = (@MOSTYPE = "PMOS")
    method gummel carriers = 1 holes

else
    method gummel carriers = 1 electrons

if.end
```

4.16.9: LOOP, L.END and L.MODIFY

These three commands together provide the standard looping functionality.

Syntax

```
loop          steps = <expr>  [print]

l.end          [break]

l.modify       [level = <expr>] [steps = <expr>] [next | break] [print]
```

Description

Every loop statement must have a corresponding l.end statement. All the commands between these two statements are executed repeatedly for the number of times given in the steps clause of the loop command. If you specify the print keyword, the values of all user-defined variables that vary under the control of the loop will print every time they change. If you specify the break keyword in the l.end statement, the loop will exit on its first iteration regardless of the value of steps.

Loops can of course be nested with each other and with IF blocks. When an l.end statement is encountered, it is associated with the most recent loop statement.

The `l.modify` statement changes the behavior of the current loop or one within which it is nested. You specify the level of the loop you wish to modify using the `level` clause, which is described in Section 4.16.3: “ASSIGN”. Without this clause, the current loop is assumed. You use the `steps` clause to change the number of times the loop will be executed. A value less than or equal to the current loop iteration count is acceptable and simply results in the loop exiting at the end of the current iteration.

The `break` keyword causes the loop to exit immediately.

The `next` keyword causes the loop to abandon the current iteration and to begin the next without executing any statements between the `l.modify` and the relevant `l.end` statements.

The `print` command switches on the printing of user-defined variables as described above.

Example

```
loop steps=3

    assign name=param1 print n.value = 1 delta = 1

loop steps=3
    assign name=param2 print n.value = 1 delta = 1
l.end

l.end
```

4.16.10: MASK

Defines the position of the process flow where photoresist or barrier material is added with the use of the MASKVIEWS IC layout interface.

Syntax

```
mask name="maskname"[misalign=<misalignment>/
|bias=<bias>|delta_cd= <delta_cd>/
|shrink=<shrink>|reverse|optolith]
```

Description

Mask is used to interface to Silvaco’s general purpose layout editor MASKVIEWS. The `mask` statement defines the location where photoresist is deposited in the flow of processing events. The etched pattern is dependent on the MASKVIEWS cutline file, which must be loaded into DECKBUILD.

Name specifies the name of the layer that defines the photoresist patterning. This name must correspond to a mask level name contained in the MASKVIEWS cutline file loaded into DECKBUILD.

Bias and delta_cd increase or decrease the width of the deposited mask. For positive masks, a positive `delta.bias` decreases the etched hole(s) in the mask.

Misalignment shifts the entire specified mask left and right. Negative misalignment values shift the mask left, positive values right.

Shrink reduces the size of the specified layer by the ratio specified.

Reverse specifies that the mask polarity should be reversed or that negative type photoresist should be modeled.

Optolith specifies that the loaded MASKVIEWS cutline is from an **Optolith** layout. Therefore, **Optolith** syntax (layout commands) is used to define the photoresist pattern.

Examples

The `delta` value can be used to vary the Critical Dimension (CD) of the specified layer. The value operates on an edge-by-edge basis. For example, for an IC layout with a 1.0, micron wide "poly" the statement:

```
mask name="poly" delta=-0.1
```

creates a drawn poly length of 0.8 microns, meaning that 0.1 have been removed from each poly edge.

The `bias` command option performs the same operation as the `delta` command. This can be used globally to edit the bias of each layer. The `bias` command can be used with `delta`, such that the real value for CD reduction is the sum of the `delta` and `bias` values, per edge. For example, if an IC layout with 1.2 micron CD's is streamed-in from GDS2, and the final etch, then the final etch profile is known to be 0.9 microns due to a combination of biasing, photo-exposure, and over etch, then the offset is required to be constant. This is where the `bias` command can be used.

```
mask name="poly" bias=-0.15
```

In other words, 1.2 microns-0.9 microns=-0.3 microns =2(-0.15) microns, or -0.15 microns per edge.

Further experimentation might be required in addition to the fixed bias. This is where the `delta` command can be used. In this example:

```
mask name="poly" bias=-0.15 bias=-0.15 delta=-0.025
```

This simulates a true experiment in terms of CD variation.

The `misalign` command is used to offset a layer with respect to other layers. For example

```
mask name="poly" bias=-0.15 misalign=-0.1
```

causes the poly layer to be offset to the left by 0.1 microns.

The `shrink` command is used to reduce the size of all edges in the specified layer. For example, the statement below will reduce the layer edges by 50 percent.

```
mask name="poly" shrink=0.5
```

Misalignment and CD Experimentation

It is often necessary to experiment with either misalignment or the CDs of a layer. The MASKVIEWS-DECKBUILD interface supports this level of experimentation. DECKBUILD can be used to experiment with the cutline generated by MASKVIEWS. Each mask statement can be used to alter the cutline. The underlying mesh used by ATHENA is not changed with mask experimentation commands. VWF can be used to split on these values to generate RSM's relating to mask experimentation.

4.16.11: MASKVIEWS

Plots a layout file

Syntax

```
maskviews <layout file>
```

Description

This statement starts the MASKVIEWS layout editor and load the supplied layout file. If no layout file is specified, MASKVIEWS is invoked with no data.

Examples

This statement plots a layout file (which should be in the current directory).

```
maskviews layout.lay
```

See

Chapter 10: “MaskViews”.

4.16.12: SET

Sets the value of a user-defined variable or clear all existing

Syntax

```
set <variable>=[ <value> | <expr> ] [nominal]
               =[ max (<expr>, <expr>) ][nominal]
               =[ min(<expr>, <expr>) ][nominal]
               =[ ave(<expr>, <expr>) ][nominal]
               =[ log(<expr>) ][nominal]
               =[ log10(<expr>) ][nominal]
               =[ sqrt (<expr>) ][nominal]
               =[ abs(<expr>) ][nominal]
               =[ exp(<expr>) ][nominal]
               =[ atan(<expr>) ][nominal]

set clear
```

Description

The `set` command is used to set the value of a user-defined variable. The value can later be substituted using `$`-substitution, which replaces the variable name with its value when the variable is preceded by a dollar sign `'$'` or by the at sign `'@'`.

variable is a user-defined variable name. It may contain spaces or algebraic operators (+, -, *, /, ^) if specified with quotes.

expr is an algebraic expression consisting of numeric constants, `$`-substituted variables, algebraic operators (+, -, *, /, ^), and or the built-in functions shown.

`set` commands can be used in conjunction with extracted values.

If a `$`-variable is being substituted and if a standard `DECKBUILD` variable is not discovered, it is assumed to be a user-defined environment variable.

Synonyms for SET

We have introduced synonyms for the `SET` syntax to provide improved compatability with other products. Both of the following syntaxes are valid.

```
[Assign|assign] NAME=<variable> N.VALUE=[<value>
                                     <expr>
                                     max  (<expr>, <expr>)
                                     min  (<expr>, <expr>)
                                     ave   (<expr>, <expr>)
                                     log   (<expr>)
                                     log10 (<expr>)
```

```

sqrt  (<expr>)      |
abs   (<expr>)      |
exp   (<expr>)      |
atan  (<expr>)
]

```

```

define <variable> [<value>
                  |
                  <expr>
                  |
                  max  (<expr>, <expr>) |
                  min  (<expr>, <expr>) |
                  ave   (<expr>, <expr>) |
                  log   (<expr>)         |
                  log10 (<expr>)         |
                  sqrt  (<expr>)         |
                  abs   (<expr>)         |
                  exp   (<expr>)         |
                  atan  (<expr>)
                  ]

```

These will assign the chosen value or expression to the variable in the same way as the existing SET syntax. To reiterate, you can substitute the value later using $\$$ -substitution, which replaces the variable name with its value when the variable is preceded either by a dollar sign '\$', or by the at sign '@'.

Examples

The following statement uses spaces and algebraic operators in the variable name.

```

set "dose+engery"=1
set "my value"=2.
set temp=1000

```

These statements show how to set variables and how to substitute with each other and in simulator syntax.

```

set time=30
set temp=1000
set press=1.0
set env="nitro"
set pi=3.1415
set "pi*2" = 2*$pi
diffuse time=$time temp=$temp press=$press $env hcl="$pi*2"

```

The following statements extract the thickness of the top layer of oxide in a structure and etch back that thickness plus 0.05 micron.

```
extract name="oxide thickness" thickness oxide
set etch_thickness = ("oxide thickness"*10000) + 0.05
etch oxide dry thickness=$etch_thickness
```

Note: The thickness is measured in angstroms, so it is converted to microns first.

Variable names that contain spaces (generated by `extract` statements) must be quoted for `$-` substitution, and the ``` must precede the quoted string as shown.

For variable names with no spaces, quotes are optional.

The following statement will remove all existing variables.

```
set clear
```

The statements below show the use of the nominal flag.

```
extract name="oxide thickness" "oxide thickness" thickness_bad_syntax oxide
set "oxide thickness" = 0.5 nominal
etch oxide dry thickness=$oxide thickness
```

For this example, if the `extract` statement was successful, the value of `"oxide thickness"` would be set. Therefore, the nominal `set` statement would be ignored. But the `extract` syntax is incorrect, so the `extract` statement never creates the result variable and `"oxide thickness"` is set by the nominal `set` statement to 0.5.

4.16.13: SOURCE

Enables simulation commands to be executed from an external file

Syntax

```
SOURCE file
```

Description

The `SOURCE` statement enables simulation commands to be executed from an external file. The named file is read and placed in `DECKBUILD`'s input buffer and is executed as if it were part of the input deck.

`file` is the name of a file that contains any valid simulator syntax or `DECKBUILD` statements, such as `extract` and `set`. The sourced file may source other files. If the file name does not begin with `'/'`, then it is assumed to be in the current directory.

Examples

The file to be sourced may contain part of an input deck including commands from any simulator. The following input deck fragment will perform a diffusion, access the file `include_file` for further commands, then revert back to the deposition step.

```
etch oxide all
#
# Source an external file
# Return to the input deck
implant bf2 dose=1.0e12 energy=35 pearson
```

```
include_file contains these statements:
#gate oxide grown here:-
diffus time=10 temp=900 dryo2 press=1.00 hcl%=3
```

The runtime output from this fragment will appear as:

```
ATHENA> etch oxide all
ATHENA> #
ATHENA> # Source an external file
ATHENA> source include_file
ATHENA> #gate oxide grown here:-
ATHENA> diffus time=10 temp=900 dryo2 press=1.00 hcl%=3
Solving time(sec.)      0 +      0.01      100%, np 106
Solving time(sec.)      0.01 +    0.173987    1739.87%, np 106
Solving time(sec.)    0.183987 +    0.187665    107.861%, np 106
*
Solving time(sec.)    0.371653 +    0.628347    334.823%, np 106
Solving time(sec.)      1 +      0.1      15.9148%, np 106
Solving time(sec.)      1.1 +     3.1396     3139.6%, np 106
Solving time(sec.)      4.2396 +    19.1813    610.947%, np 106
Solving time(sec.)     23.4209 +    93.4041    486.955%, np 106
Solving time(sec.)     116.825 +      150     160.593%, np 104
Solving time(sec.)     266.825 +      150      100%, np 104
Solving time(sec.)     416.825 +      150      100%, np 104
Solving time(sec.)     566.825 +    33.1751    22.1167%, np 104
ATHENA ># Return to the input deck
ATHENA> implant bf2 dose=1.0e12 energy=35 pearson
```

4.16.14: STMT

Enables you to define variables that change under the control of loops.

Syntax

```
stmt <parameters>
```

Where

```
<parameters> -> <parameter> | <parameter> <parameters>
```

```
<parameter> -> <variable> = <initial> [ : [ + | * ] <change> [ : <level> ] ]
```

That is, a `stmt` command must carry at least one parameter and may carry many (independent) parameters.

Description

This is effectively a shorthand for part of the ASSIGN statement.

The <initial>, <change> and <level> terms are all numerical expressions.

The value of the variable is re-evaluated **every** time the STMT command is encountered. If no arithmetical operator is specified or if the '+' sign appears explicitly, then addition is understood and the variable is re-evaluated as

$$\langle \text{initial} \rangle + \langle \text{change} \rangle * (\text{count} - 1)$$

where count is the current iteration count of the loop with level <level>.

If the multiplication operator ('*') appears, the variable is re-evaluated as

$$\langle \text{initial} \rangle * \langle \text{change} \rangle ^ (\text{count} - 1)$$

where the caret (^) stands for the power operator and is counted as before.

The <change> term defaults to 0 in the addition case and 1 in the multiplication case. The <level> term defaults to the current loop level. This means that if you only specify <initial>, the variable will be a constant.

Examples

1. In this example param1 will take the values 1, 2, 3, 4 and 5.

```
loop steps=5 print
    stmt param1=1:1
1.end
```

2. In this example param1 will take the values 1, 2, 4, 8 and 16.

```
loop steps=5 print
    stmt param1=1:*2
1.end
```

4.16.15: SYSTEM

Allows DECKBUILD to execute UNIX system commands within a simulation deck.

Syntax

```
SYSTEM <UNIX command>
```

Description

The SYSTEM command allows you to execute shell scripts or perform other UNIX tasks directly from the simulation deck. The command is blocking, meaning that the simulation does not continue until the SYSTEM command has finished execution.

To use this feature, enable the SYSTEM commands in the Main Control Options Popup (see Figure 4-12). To enable system commands for VWF AUTOMATION TOOLS, set the environment variable DB_SYSTEM_OPTION to any value.

Examples

```
system rm history*.str
```

Note: Redirection of the system command output (i.e., `system ls * .in > file.out`) cannot be achieved as the output is already redirected by DECKBUILD.

4.16.16: TONYPLOT

Plots a file

Syntax

```
tonyplot -args
```

Description

This statement causes DECKBUILD to save a temporary file from the current simulator and start up TONYPLOT with that file loaded. The temporary file is removed when TONYPLOT exits.

-args, if specified, are passed directly to TONYPLOT (as if invoked from the command line). If any of -st, -da, or -over and a file name is specified, DECKBUILD uses the named file instead of saving and plotting the current structure.

DECKBUILD also detects if the structure to be plotted is 3D and use TONYPLOT3D if required.

Examples

This statement saves the current file and starts TONYPLOT.

```
tonyplot
```

This statement plots a file (which should be in the current directory).

```
tonyplot -st well.str
```

See

Section 4.5: “Main Control”.

4.17: Environment Variables

S_EXAMPLES specifies the location of the Silvaco standard examples and overrides the default setting.

<simulator>_ARG overrides the default simulator arguments set in the Simulator Properties popups. The variable below would setup DECKBUILD to start version 1.0.0.A of ATHENA by default.

```
setenv ATHENA_ARG "athena -V 1.0.0.A"
```

<simulator>_HOST overrides the default simulator host setting in the Simulator Properties popups. The variable below would setup DECKBUILD to start ATLAS on hostname `sgi3` by default.

```
setenv ATLAS_HOST sgi3
```

DB_MESH_DIR sets the template directory used in the ATHENA Adaptive Meshing popup. The template files are read and displayed from the directory specified by this variable.

DB_SYSTEM_OPTION enables system commands for use in DECKBUILD and VWF AUTOMATION TOOLS.

NICE_ARG sets the simulator nice value for use in DECKBUILD and VWF AUTOMATION TOOLS.

DB_REMOTE_DIR sets the remote simulation tmp directory, which must be mounted on the host executing DECKBUILD and the host executing the simulator

DB_REMOTE_CMD specifies the remote shell command to be used for remote simulation.

DB_REMOTE_STRIP specifies the automount prefix (usually `/tmp_mnt`) to be removed from paths for remote simulation.

4.18: Error Messages

4.18.1: Text Subwindow Error Messages

The error message “Insertion failed” may occasionally pop up when building very large decks. This means that so many edits were made to the deck that the text subwindow can’t handle any more changes. This only happens when building a deck that has not been saved to a file yet. Otherwise, the text editor automatically saves the changes to the file as necessary.

The cure and the prevention are the same: if this error occurs, simply save the deck to a file. An alternate preventative measure is to add a line of the form:

```
text.maxDocumentSize: N
```

to the `.Xdefaults` file, where `N` is the number of bytes allowed before `textedit` saves the file (2000 by default). Try using `N` set to 20000.

4.18.2: TTY Subwindow Error Messages

The `tty` subwindow, like the text subwindow, has a limit on how big it can grow before an error occurs. In rare circumstances, it is possible to overflow the `tty` subwindow. But it takes a great deal of simulation output to do it (many thousands of lines). If an error message such as “`pty: insertion failed`” appears, either clear the contents of the `tty` subwindow, or turn scrolling off (because the log is only saved if scrolling is turned on). If this error comes up repeatedly, the best solution is to disable scrolling. Disable scrolling by invoking the **tty** menu in the `tty` subwindow and selecting **Disable Scrolling**.

5.1: Overview

DECKBUILD has a built-in extraction language that allows measurement of physical and electrical properties in a simulated device. The result of all extract expressions is either a single value (such as X_j for process or V_t for device), or a two-dimensional curve (such as concentration versus depth for process or gate voltage versus drain current for device).

EXTRACT forms a “function calculator” that allows you to combine and manipulate values or entire curves quickly and easily. You can create your own, customized expressions, or choose from a number of standard routines provided for the process and device simulators. You can take one of the standard expressions and modify it as appropriate to suit your needs. EXTRACT also has variable substitution capability so that you can use the results of previous `extract` commands.

EXTRACT has two built-in 1D device simulators, QUICKMOS and QUICKBIP, for specialized cases of MOS and bipolar electrical measurement. Both QUICKMOS and QUICKBIP run directly from the results of process simulation for fast, easy and accurate device simulation.

5.2: Process Extraction

DECKBUILD's process extraction window is shown below (Figure 5-1).

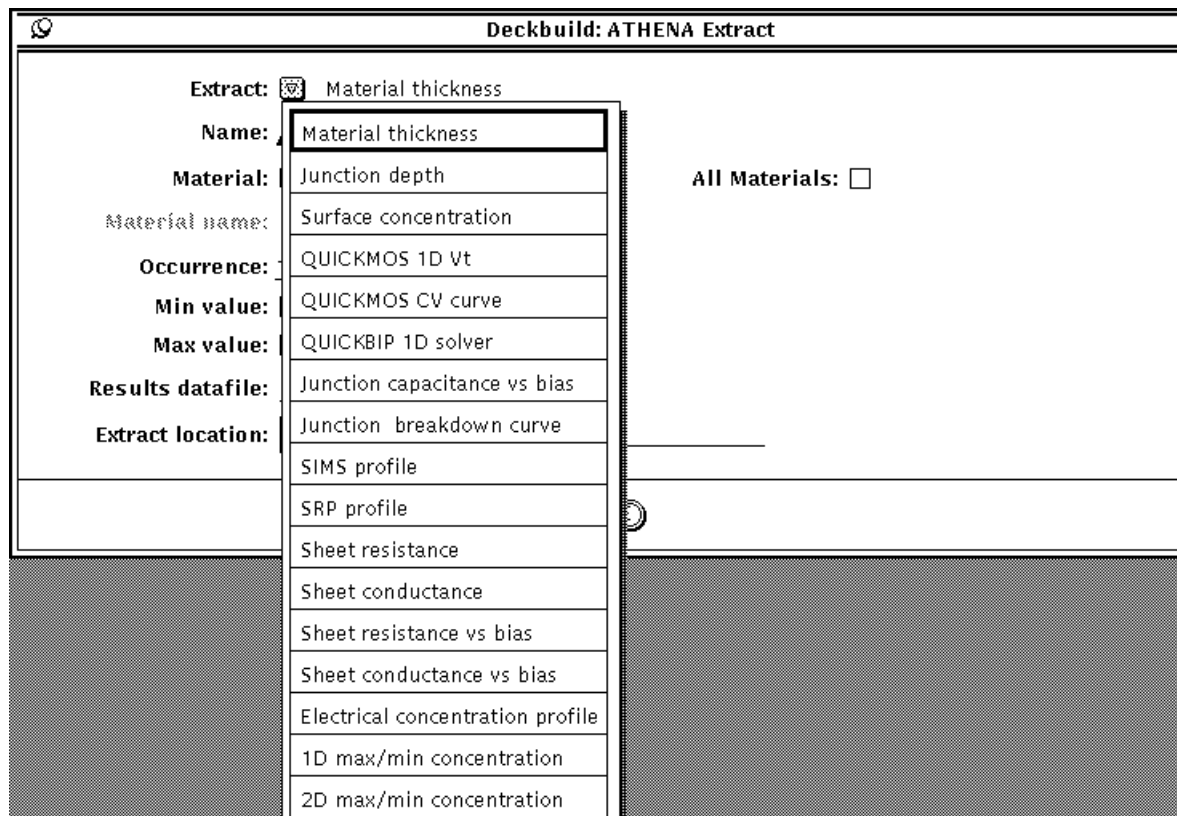


Figure 5-1: Process Extraction

You may use this window to look at the following:

- **Material thickness** measures the thickness of the nth occurrence of any material or all materials in the structure.
- **Junction depth** measures the depth of any junction occurrence in the nth occurrence of any material.
- **Surface concentration** measures the surface concentration of any dopant, or net dopant, in the nth occurrence of any material.
- **QUICKMOS 1D Vt** calculates the one-dimensional threshold voltage of a MOS cross section using the built-in QUICKMOS 1D device simulator. The gate voltage range defaults between 0 to 5 Volts but can be specified as required. The substrate can also be fixed at any bias. Qss and device temperature values may also be specified.
- **QUICKMOS CV curve** creates a CV curve of a MOS cross section using QUICKMOS. This shows capacitance as a function of either gate voltage or substrate voltage with the other terminal held at any fixed bias. Qss and device temperature values may also be specified.
- **QUICKBIP 1D solver** measures any of 22 BJT Gummel-Poon parameters, plus any forward or reverse IV curve. See the Section 5.9: "QUICKBIP Bipolar Extract" for more information and examples.
- **Junction capacitance versus bias** calculates the junction capacitance of a specified p-n junction within any region as a function of applied bias to that region. Qss and device temperature values can also be specified.

- **Junction breakdown curve** calculates the electron or hole ionization integral of any region as a function of applied bias to that region. This calculation uses the Selberherr impact ionization model. (see “Impact” command section and “Impact Ionization” physics section within the ATLAS manual). You can modify the Selberherr model default values and specify Q_{ss} and device temperature values.
- **SIMS profile** calculates the concentration profile of a dopant in a material layer.
- **SRP profile** calculates the SRP (Spreading Resistance Profile) in a silicon layer.
- **Sheet resistance and sheet conductance** calculates the sheet resistance or conductance of any p-n region in any layer in an arbitrary structure. You can specify the bias of any region in any layer, the Q_{ss} of any material interface and the device temperature. A flag for carrier freezeout calculations can also be set (see “Incomplete Ionization Of Impurities” physics section within the ATLAS manual).
- **Sheet resistance and sheet conductance versus bias** calculates the sheet resistance or conductance of one or more regions as a function of applied bias to any region. Q_{ss} and device temperature values can also be specified.
- **Electrical concentration profiles** measures electrical distributions versus depth. You can also specify the bias of any region in any layer and the Q_{ss} of any material interface. The device temperature can also be set to the required value. The following distributions are calculated:
 - electrons
 - holes
 - electron quasi-fermi level
 - hole quasi-fermi level
 - intrinsic concentration
 - potential
 - electron mobility
 - hole mobility
 - electric field
 - conductivity
- **1D maximum/minimum concentration** measures the peak or minimum concentration of any dopant or net dopant, for a specified 1D cutline, in the nth occurrence of any material or all materials, and also within any junction-defined.
- **2D maximum/minimum concentration** measures the peak or minimum concentration of any dopant or net dopant, for the whole 2D structure or within a specified area, in any material or all materials, and also within any junction-defined. The actual xy coordinates of the maximum or minimum concentration can also be retrieved.
- **2D material region boundary** returns the maximum or minimum boundary of the selected material region for either X or Y axis. Therefore, the outer boundaries of any material region can be extracted.
- **2D concentration area** integrates specified concentration of any dopant or net dopant for whole 2d structure or within a specified location.
- **2D maximum concentration file (CCD)** creates a **Data Format** file with the XY coordinates and the actual values of the maximum concentrations stepping across the structure. This file can be loaded into TONYPLOT when in **-ccd** mode to show a line of maximum concentration across a device.
- **ED tree** creates one branch of a **Smile** plot or **ED tree** from multiple **Defocus** distance against **Critical Dimension (CD)** plots created for a sweep of **Dose** values by OPTOLITH. These plots are all written in a single Data format file.
- **Elapsed time** extracts time stamps from a specified start time at any point in a simulation. You can reset the start time as required.

Note: This extraction is not CPU time.

The built-in 1D Poisson device simulator is used to calculate sheet resistance and conductance and the electrical concentration profiles.

With the exception of 2D extractions, all the process extraction routines are available from both 1D and 2D process simulators. In the case of the 2D simulators, a cross section x or y value or region name (used in conjunction with MASKVIEWS) determines the 1D section to use.

Note: An error will be returned for attempted extractions on 3D structure files.

5.2.1: Entering a Process Extraction Statement

To place an `extract` statement in your process deck, select **Commands**→**Extract...** The **Extraction** popup appears. The popup for ATHENA is shown in Figure 5-2.

Figure 5-2: The ATHENA Extraction Popup

Choose the extract routine you want by activating a choice on the **Extract** setting. The popup changes size and display different items, depending on which routine you choose. Then, enter or choose the desired information for each item on the popup. An extract name is always be required. Optionally, enter the minimum or maximum desired cutoff values by checking **Min value** or **Max value** and entering a value. By default, all extract results are written to a file named `results.final`. But using the **Results datafile** field allows you to specify the results file for each individual extract statement. Material and impurity names are selected using a **Chooser** (Figure 5-3).

Figure 5-3: Material Chooser Popup

If the required option is not present in the default setting, select the **User** filter to search for other materials/impurities. The **Hide Worksheet Result** setting specifies that this extract should not be displayed in the VWF INTERACTIVE TOOLS worksheet. This prevents extracts used for calculation purposes only from cluttering the worksheet results.

Finally, place the text caret at the desired point in the deck and click on the **WRITE** button. The extract syntax is written to the deck.

5.2.2: Extracting a Curve

Some of the process extraction statements create a two-dimensional curve as a result, rather than a single value. For instance, `extract` constructs a data set of concentration versus depth for the SIMS, SRP, and electrical quantities distributions. You can use the resulting 2-D curve for measurement and testing and as a target on the OPTIMIZER worksheet so that you can optimize against 2-D curves.

EXTRACT provides several additional options to 2-D curve support: axis layout, axis attributes, optional computation of area under the curve, and optional outfile. These options are the same regardless what type of curve (for instance, QUICKMOS CV and SIMS profile) you are extracting.

The ATHENA Extract popup showing the SIMS Profile is shown in Figure 5-4.

Deckbuild: ATHENA Extract

Extract: ☒ SIMS profile

Name: _____

Material: Silicon All Materials: ☐

Material name: _____

Occurrence: 1 ☐ 10

Impurity: Net Doping

X vs Y axis: Depth vs Concentration Compute curve area: ☐

X axis attributes:

Y axis attributes:

X axis bounds: ☐

X axis start: _____ X axis end: _____

Store X/Y datafile: Yes ☒ No ☐ Filename: extract.dat

Results datafile: results.final Hide worksheet result: ☐

Extract location: X ☒ Y ☐ Region name ☐ Value: _____

Figure 5-4: ATHENA Extract Popup with SIMS Profile

The following options are available:

- **X vs Y axis** determines the x and y axes of the resulting profile curve. The default (which should always be used unless you plan to customize the resulting extract expression) is that the x axis is depth into the material, and the y axis is the concentration.
- **Curve X axis bounds** specifies whether to create the curve for the whole X axis or for only a required section. If selected, **X axis value** fields become active, enter values in the same units as the resulting curve. This is useful for extracting local maxima and minima.
- **X axis attributes** and **Y axis attributes** allows you to modify the data values on each axis independently. To compute net concentration versus depth, you can select **abs** on the y axis (concentration), and select nothing on the x axis (depth). **abs** is always evaluated before taking the log or square root of the data.
- **Curve X axis bounds** specifies whether to create the curve for the whole X axis or for only a required section. If selected, **X axis value** fields become active, enter values in the same units as the resulting curve. This is useful for extracting local maxima and minima.
- **Store X/Y datafile** stores an output file in TONYPLOT data format if set to **Yes**. You can plot the data file in TONYPLOT using the `-da` option. You can also read the data file directly into the OPTIMIZER worksheet as a target if desired.
- **Compute curve area** computes the area under the curve. When checked, it causes several other items to become active:
- **Area X axis bounds** tells EXTRACT whether to integrate the area under the curve along its entire length or just for a bounded portion of the X axis. If you select **Bounded**, then **X axis start** and **X axis stop** become active. Enter **start** and **stop** values in the same units as the resulting curve.

To construct the 2-D curve, set each item on the popup in turn and click on **WRITE**.

Depth is always computed as distance from the top of the selected material layer and occurrence. Depth starts from 0 and increases through the material.

5.3: Customized Extract Statements

In addition to the simple curve primitives shown on the popup, you can edit the input deck directly to make customized curves. Examples include extracting maxima and minima on the curve, combining axes using a function definition, looking at slopes of tangent lines, intercepts of sloped lines. The EXTRACT syntax is described below, followed by examples of process extraction. See the examples listed under Section 5.4: “Device Extraction” for more information.

5.3.1: Extract Syntax

Text inside matching pairs of /* and */ delimiters are comments. These are used to clarify the meaning of the syntax and also as definitions for the most primitive types, such as <QSTRING>.

The backslash character (\) at the end of a line indicates a continuation line.

Many of the optional parameters (the ones enclosed in square brackets) have default values. Some of these defaults are given immediately after they appear. Others appear in more than one place and so are collected at the end.

Description

<EXTRACT_STATEMENT> :

<EXTRACT_SINGLE_LINE_GENERAL>

<EXTRACT_MULTIPLE_LINE_GENERAL>

<EXTRACT_2D_MAX_MIN_CONC>

<EXTRACT_TIME>

<EXTRACT_SIMPLE>

<EXTRACT_SINGLE_LINE_GENERAL> :

[extract init infile=QSTRING]

/* In default of the above line, a temporary structure file representing the current state of the device will be constructed. */

extract [name=<QSTRING>] <EXTRACT_SINGLE_LINE_PARTICULAR> \

[datafile=<QSTRING>] [hide]

<EXTRACT_MULTIPLE_LINE_GENERAL> :

[extract init infile=<QSTRING>]

/* In default of the above line, a temporary structure file representing the current state of the device will be constructed. */

extract start <EXTRACT_MULTIPLE_LINE_SETUP_N>

[extract cont <EXTRACT_MULTIPLE_LINE_SETUP_N> ...]

```
/* zero or more instances of the extract cont line may appear. */

extract done [name=<QSTRING>] <EXTRACT_MULTIPLE_LINE_DONE_N>          \
           [datafile=<QSTRING>] [hide]

/* There are five pairs of definitions for<EXTRACT_MULTIPLE_LINE_SETUP_N>
   and <EXTRACT_MULTIPLE_LINE_DONE_N>, with N replaced by 1, 2, 3, 4 or 5.
   Elements from different pairs (ones with different values of N) must
   NOT appear in the same statement. */

<EXTRACT_2D_MAX_MIN_CONC> :

    [extract init infile=<QSTRING>]

/* In default of the above line, a temporary structure file representing
   the current state of the device will be constructed. */

extract [name=<QSTRING>] 2d.max.conc | 2d.min.conc [interpolate]      \
[<IMPURITY>] [<MATERIAL>] [mat.occno=<EXPR>]                        \
[min.v=<EXPR>][max.v=<EXPR>]                                          \
[x.max=<EXPR> x.min=<EXPR> y.max=<EXPR> y.min=<EXPR> |                \
y.max=<EXPR> y.min=<EXPR> region=<QSTRING>]                          \
[datafile=<QSTRING>] [hide]

[extract [x_pos_name=<QSTRING>] x.pos

    extract [y_pos_name=<QSTRING>] y.pos]

/* x_pos_name and y_pos_name will default to the name in the main extract
   statement, with "X position" and "Y position" appended respectively. */

<EXTRACT_TIME> :

    extract [name=<QSTRING>] clock.time [start_time=<EXPR>]          \
[datafile=<QSTRING>]

<EXTRACT_SIMPLE> :

    extract [name=<QSTRING>] <EXPR> [datafile=<QSTRING>]

<EXTRACT_SINGLE_LINE_PARTICULAR> :

    <CURVE_FUNC> (<CURVE_SINGLE_LINE>) [outfile=<QSTRING>]
```

<EXTRACT_MULTIPLE_LINE_DONE_5>

```
thickness [min.v=<EXPR>] [max.v=<EXPR>] [<MATERIAL>] [mat.occno=<EXPR>] \
[x.val=<EXPR> | y.val=<EXPR> | region=<QSTRING>]
```

```
xj [min.v=<EXPR>] [max.v=<EXPR>] [<MATERIAL>] [mat.occno=<EXPR>] \
[x.val=<EXPR> | y.val=<EXPR> | region=<QSTRING>] [junc.occno=<EXPR>]
```

```
surf.conc [min.v=<EXPR>] [max.v=<EXPR>] [<MATERIAL>] [mat.occno=<EXPR>] \
[x.val=<EXPR> | y.val=<EXPR> | region=<QSTRING>] [<IMPURITY>]
```

```
ldvt [ptype | ntype] [min.v=<EXPR>] [max.v=<EXPR>] \
[bias=<EXPR>] [bias.start] [bias.stop=<EXPR>] [bias.step=<EXPR>] \
[vb=<EXPR>] [temp.val=expr] [soi] [qss=<EXPR>] [workfunc=<EXPR>] \
[x.val=<EXPR> | y.val=<EXPR> | region=<QSTRING>]
```

/* Default values : ptype, vb=0.0, qss=0 */

```
max.conc | min.conc [<IMPURITY>] [<MATERIAL>] [mat.occno=<EXPR>] \
[region.occno=<EXPR>]
```

/* region.occno will default to all regions. */

```
2d.conc.file [<IMPURITY>] [<MATERIAL>] [mat.occno=<EXPR>] \
[x.max=<EXPR> x.min=<EXPR> y.max=<EXPR> y.min=<EXPR>]
```

```
max.conc.file | min.conc.file [<IMPURITY>] [<MATERIAL>] [xstep=<EXPR>] \
[x.max=<EXPR> x.min=<EXPR> y.max=<EXPR> y.min=<EXPR>]
```

```
max.bound | min.bound x.val=<EXPR> | y.val=<EXPR> \
[min.v=<EXPR>] [max.v=<EXPR>] [MATERIAL] [mat.occno=<EXPR>]
```

```
max.bound | min.bound x.pos | y.pos xval=<EXPR> y.val=<EXPR> \
[MATERIAL] [min.v=<EXPR>] [max.v=<EXPR>]
```

```
2d.area [<IMPURITY>] [x.step=<EXPR>] [min.v=<EXPR>] [max.v=<EXPR>] \
[x.max=<EXPR> x.min=<EXPR> y.max=<EXPR> y.min=<EXPR> |
y.max=<EXPR> y.min=<EXPR> region=<QSTRING>]
```

```
/* Default value : x.step = 10% of device size */
```

<CURVE_FUNC> (<CURVE_ARG>) :

<CURVE_ARG>

min (<CURVE_ARG>)

/* Returns min y val for curve. */

max (<CURVE_ARG>)

/* Returns max y val for curve. */

ave (<CURVE_ARG>)

/* Returns average value for curve. */

slope | xintercept | yintercept (maxslope | minslope (<CURVE_ARG>))

/* Takes the tangent to the curve with either the least or the greatest
slope and returns either the slope of this tangent, or its x intercept,
or its y intercept. */

area from (<CURVE_ARG>) [where x.min=<EXPR> and x.max=<EXPR>]

/* Determines the area under the specified curve between the x limits
defined by the min and max expressions. */

x.val from (<CURVE_ARG>) where y.val=<EXPR> [and val.occno=<EXPR>]

y.val from (<CURVE_ARG>) where x.val=<EXPR> [and val.occno=<EXPR>]

/* Determines the x (or y) ordinate on the curve where the corresponding
y (or x) ordinate is equal to the constant expression for the occurrence
specified. Linear interpolation is used between points on the curve.*/

grad from (<CURVE_ARG>) where x.val=<EXPR> | y.val=<EXPR>

```

/* Determines the gradient at the first x (or y) ordinate on the curve
   where the corresponding y (or x) value is equal to the consent
   expression. Linear interpolation is used between points on the curve.*/

<CURVE_SINGLE_LINE> :

curve (<AXIS_FUNC> (bias),                                     \
      <AXIS_FUNC> (ldcapacitance [vg=<EXPR>] [vb=<EXPR>]      \
        [bias.ramp=vg|vb]\                                   \
        [bias.step=<EXPR>] [bias.start=<EXPR>]                \
        [bias.stop=<EXPR>][temp.val=expr][soi][qss=<EXPR>]    \
        [workfunc=<EXPR>]                                     \
        [x.val=<EXPR> | y.val=<EXPR> | region=<QSTRING>]       \
      )                                                       \
      [, xmin=<EXPR> xmax=<EXPR>]                               \
    )

/* Default values : vg=0.0, vb=0.0, bias.ramp=vg, qss=0 */

curve (<AXIS_FUNC> (depth),                                   \
      <AXIS_FUNC> ([<IMPURITY>] [<MATERIAL>] [mat.occno=<EXPR>] \
        [x.val=<EXPR> | y.val=<EXPR> | region=<QSTRING>]         \
      )                                                         \
      [, xmin=<EXPR> xmax=<EXPR>]                               \
    )

curve (<AXIS_FUNC> (depth),                                   \
      <AXIS_FUNC> (srp                                         \
        [material="silicon"|"polysilicon"] [mat.occno=<EXPR>] \
        [x.val=<EXPR> | y.val=<EXPR> | region=<QSTRING>]         \
      )                                                         \
      [, xmin=<EXPR> xmax=<EXPR>]                               \
    )

curve (<AXIS_FUNC> (<DEV_AXIS>),                               \
      <AXIS_FUNC> (<DEV_AXIS>)                                 \
      [, xmin=<EXPR> xmax=<EXPR>]                               \
    )

deriv (<AXIS_FUNC> (<DEV_AXIS>),                              \
      <AXIS_FUNC> (<DEV_AXIS>)                                 \
      [, <INTEGER>]                                           \
    )

```

```
/* The integer is of course the nth derivative and its default value is 1.*/
```

```
edcurve (<DEFOCUS_AXIS>, <CRITICAL_DIMENSION_AXIS>, <DOSE_AXIS>, \
        dev=<EXPR> datum=<EXPR> x.step=<EXPR>)
```

```
<AXIS_FUNC> (<AXIS_ARG>) :
```

```
<AXIS_ARG>
```

```
<AXIS_ARG> + <EXPR>
<EXPR>      + <AXIS_ARG>
<AXIS_ARG> + <AXIS_ARG>
```

```
<AXIS_ARG> - <EXPR>
<EXPR>      - <AXIS_ARG>
<AXIS_ARG> - <AXIS_ARG>
```

```
<AXIS_ARG> / <EXPR>
<EXPR>      / <AXIS_ARG>
<AXIS_ARG> / <AXIS_ARG>
```

```
<AXIS_ARG> * <EXPR>
<EXPR>      * <AXIS_ARG>
<AXIS_ARG> * <AXIS_ARG>
```

```
<AXIS_ARG> ^ <EXPR>
<EXPR>      ^ <AXIS_ARG>
<AXIS_ARG> ^ <AXIS_ARG>
```

```
-<AXIS_ARG>
```

```
abs  (<AXIS_ARG>)
log  (<AXIS_ARG>)
log10(<AXIS_ARG>)
sqrt (<AXIS_ARG>)
atan (<AXIS_ARG>)
```

```
<EXPR> :
```

```
<NUMBER>
```

```
$variable | $"variable"
```

```
/* deckbuild set variable, see section 5.8.2: Variable Substitution */
```

```
expr + expr
expr - expr
```



```

    expr / expr
    expr * expr

    (expr)
    -expr

<EXTRACT_MULTIPLE_LINE_SETUP_1> :

    <EXTRACT_MULTIPLE_LINE_SETUP_A>

<EXTRACT_MULTIPLE_LINE_DONE_1> :

    <CURVE_FUNC> (<CURVE_MULTIPLE_LINE_1>) [outfile=<QSTRING>]

<CURVE_MULTIPLE_LINE_1> :

    curve (<AXIS_FUNC> (bias),
        <AXIS_FUNC> (ldjunc.cap [<MATERIAL>] [mat.occno=<EXPR>]
            [region.occno=<EXPR>] [junc.occno=<EXPR>]
            [temp.val=<EXPR>][soi][qss=<EXPR>][workfunc=<EXPR>]
            [y.val=<EXPR>|x.val=<EXPR>|region=<QSTRING>]
            )
        [, xmin=<EXPR> xmax=<EXPR>]
    )

<EXTRACT_MULTIPLE_LINE_SETUP_2> :

    <EXTRACT_MULTIPLE_LINE_SETUP_A>

<EXTRACT_MULTIPLE_LINE_DONE_2> :

    <CURVE_FUNC> (<CURVE_MULTIPLE_LINE_2>) [outfile=<QSTRING>]

<CURVE_MULTIPLE_LINE_2>:

    curve (<AXIS_FUNC> (bias),
        <AXIS_FUNC> ([p.ion | n.ion] [<MATERIAL>] [mat.occno=<EXPR>]
            [region.occno=<EXPR>] [junc.occno=<EXPR>]
            [temp.val=<EXPR>][soi][qss=<EXPR>][workfunc=<EXPR>]
            [y.val=<EXPR> | x.val=<EXPR> | region=<QSTRING>]
            [an1=<EXPR>] [an2=<EXPR>] [bn1=<EXPR>] [bn2=<EXPR>]
            [ap1=<EXPR>] [ap2=<EXPR>] [bp1=<EXPR>] [bp2=<EXPR>]
            [betan=<EXPR>][betap=<EXPR>][egran=<EXPR>]
            )
        [, xmin=<EXPR> xmax=<EXPR>]
    )

/* Default value : p.ion

    an1=2.03e5, an2=7.03e5,  bn1=1.231e6, bn2=1.231e6,

```

```

ap1=6.71e5, ap2=1.582e6, bp1=1.693e6, bp2=2.036e6,
betan=1.0, betap=1.0, egran=4e5

```

See Appendix A5: Threshold Voltage Calculation. */

<EXTRACT_MULTIPLE_LINE_SETUP_3> :

<EXTRACT_MULTIPLE_LINE_SETUP_A> | <EXTRACT_MULTIPLE_LINE_SETUP_B>

<EXTRACT_MULTIPLE_LINE_DONE_3> :

<CURVE_FUNC> (<CURVE_MULTIPLE_LINE_3>) [outfile=<QSTRING>]

<CURVE_MULTIPLE_LINE_3> :

```

curve (<AXIS_FUNC> (bias),
      <AXIS_FUNC> (lds.sheet.res | ldp.sheet.res | ldn.sheet.res |
                    ldconduct   | ldp.conduct   | ldn.conduct   |
                    [material="silicon" | "polysilicon"]
                    [region.occno=<EXPR>] [mat.occno=<EXPR>]
                    [y.val=<EXPR> | x.val=<EXPR> | region=<QSTRING>]
                    [workfunc=<EXPR>] [soi] [semi.poly] [incomplete]
                    [temp.val=<EXPR>]
      )
      [, xmin=<EXPR> xmax=<EXPR>]
)

```

<EXTRACT_MULTIPLE_LINE_SETUP_4> :

<EXTRACT_MULTIPLE_LINE_SETUP_A> | <EXTRACT_MULTIPLE_LINE_SETUP_B>

<EXTRACT_MULTIPLE_LINE_DONE_4> :

<CURVE_FUNC> (<CURVE_MULTIPLE_LINE_4>) [outfile=<QSTRING>]

<CURVE_MULTIPLE_LINE_4> :

```

curve (<AXIS_FUNC> (bias),
      <AXIS_FUNC> (n.conc   | p.conc   | n.qfl   | p.qfl   |
                    intrinsic | potential | n.mobility | p.mobility |
                    efield   | econductivity
                    [material="silicon" | "polysilicon"]
                    [region.occno=<EXPR>] [mat.occno=<EXPR>]
                    [y.val=<EXPR> | x.val=<EXPR> | region=<QSTRING>]
                    [workfunc=<EXPR>] [soi] [semi.poly] [temp.val=<EXPR>]
      )
      [, xmin=<EXPR> xmax=<EXPR>]
)

```

<EXTRACT_MULTIPLE_LINE_DONE_5> :

```

    sheet.res|p.sheet.res|n.sheet.res|conduct|p.conduct|n.conduct      \
[material="silicon"|"polysilicon"] [region.occno=<EXPR>]                \
[mat.occno=<EXPR>]                                                       \
[y.val=<EXPR> | x.val=<EXPR> | region=<QSTRING>]                          \
[workfunc=<EXPR>] [soi] [semi.poly] [incomplete] [temp.val=<EXPR>]
```

<EXTRACT_MULTIPLE_LINE_SETUP_5> :

<EXTRACT_MULTIPLE_LINE_SETUP_A> | <EXTRACT_MULTIPLE_LINE_SETUP_B>

<EXTRACT_MULTIPLE_LINE_SETUP_A> :

```

[<MATERIAL>] [mat.occno=<EXPR>] [region.occno=<EXPR>]                \
[bias=<EXPR>] [bias.start=<EXPR>] [bias.step=<EXPR>] [bias.stop=<EXPR>] \
[y.val=<EXPR> | x.val=<EXPR> | region=<QSTRING>]
```

<EXTRACT_MULTIPLE_LINE_SETUP_B> :

```
[interface.occno=<EXPR>] [qss=<EXPR>]
```

```
/* Default value : qss=1e10 */
```

<DEV_AXIS> :

```

v."<electrode>"                /* voltage at electrode */
i."<electrode>"                /* current at electrode
c."<electrode1>" "<electrode2>" /* capacitance between
                                electrode1 and electrode2 */
g."<electrode1>" "" "<electrode2>" /* conductance between
                                electrode1 and electrode2 */
vint."<electrode>"            /* internal voltage at electrode */
time                          /* transient time */
temperature | temp            /* device temperature */
frequency | freq              /* frequency */
beam."<beam no>"               /* light intensity for specified beam */
```

s.imaginary."<Mode>"	/* imaginary component of specified "S" code*/
s.real."<Mode>"	/* real component of specified "S" code */
h.imaginary."<Mode>"	/* imaginary component of specified "H" code*/
h.real."<Mode>"	/* real component of specified "H" code */
ie."<electrode>"	/* electron current */
q."<electrode>"	/* charge */
id."<electrode>"	/* displacement current */
ireal."<electrode>"	/* real component of current */
iimag."<electrode>"	/* imaginary component of current */
ifn."<electrode>"	/* fowler nordhiem current */
ihe."<electrode>"	/* hot electron current */
ihh."<electrode>"	/* hot hole electron current */
wfd."<electrode>"	/* workfunction difference */
rl."<electrode>"	/* lumped resistance */
cl."<electrode>"	/* lumped capacitance */
ll."<electrode>"	/* lumped inductance */
vcct.node."<circuit node>"	/* circuit bias */
icct.node."<circuit node>"	/* circuit current */
rhoe."<layer>"	/* Electron sheet resistance for layer */
rhoh."<layer>"	/* Hole sheet resistance for layer */
rho."<layer>"	/* Total sheet resistance for layer */
vlayer."<layer>"	/* Bias on layer */
sm."<mode>"	/* Photon density mode */
pm."<mode>"	/* Laser power per mirror mode */
gm."<mode>"	/* Gain mode */
vcct.real."<circuit node>"	/* Real circuit bias */
vcct.imag."<circuit node>"	/* Imaginary circuit bias */

```
icct.real."<circuit node>"      /* Real circuit current */
icct.imag."<circuit node>"      /* Imaginary circuit current */
abcd.real."<mode>"              /* ABCD real parameter */
abcd.imag."<mode>"              /* ABCD imaginary parameter */
y.real."<mode>"                 /* Y real parameter */
y.imag."<mode>"                 /* Y imaginary parameter */
z.real."<mode>"                 /* Z real parameter */
z.imag."<mode>"                 /* Z imaginary parameter */
probe."<probe name>"           /* Atlas probe values */
elect."<PARAMETER>"            /* Value for specified electrical
                               parameter */
```

<PARAMETER> :

```
time
light frequency
freq
frequency
temp
temperature
current gain
unilateral power
gain frequency
max transducer power gain
luminescent power
luminescent wavelength
optical source frequency
available photo current
source photo current
```

optical wavelength
position xhole mobility
time step magnitude
time step number
total integration time
cutoff frequency
distance along line
norm intensity
integrated e- conc
integrated h+ conc
channel sheet conductance
photon energy
photon density gain
spontaneous emission rate
electron mobility
hole current generation rate
lattice temp
electric field
recombination rate
displacement current
electron conc
hole conc
electron temp
hole temp
relative permittivity
potential

<DEFOCUS_AXIS> :

da.value"DEFOCUS" | da.value"<CURVE_NUMBER>" "DEFOCUS"

<CRITICAL_DIMENSION_AXIS> :

da.value"CDs" | da.value"<CURVE_NUMBER>" "CDs"

<DOSE_AXIS> :

da.value"DOSE" | da.value"<CURVE_NUMBER>" "DOSE"

<CURVE_NUMBER> :

/* Integer specifying which curve when multiple curves are present in a
DA format file. */

<MATERIAL> :

Silicon
Oxide
SiO~2
Oxynitride
Nitride
Si~3N~4
Polysilicon
Photoresist
Barrier
Aluminum
Tungsten
Titanium
Platinum
Cobalt
Tungsten Silicide
Titanium Silicide
Platinum Silicide
Cobalt Silicide
GaAs
AlGaAs
InGaAs
SiGe
InP
6H-SiC
4H-SiC
3C-SiC
Germanium
material=<QSTRING>

<IMPURITY> :

Boron

```
Phosphorus
Arsenic
Bf2
Antimony
Silicon
Zinc
Selenium
Beryllium
Magnesium
Aluminum
Gallium
Carbon
Indium
Chromium
Germanium
impurity=<QSTRING>
```

<NUMBER> :

```
/* Real or integer value. */
```

<QSTRING> :

```
/* Quoted string, for example, "silicon". */
```

5.3.2: DEFAULTS

The following default values will be assumed.

name=<QSTRING>	: name=None
<MATERIAL>	: material="silicon"
<IMPURITY>	: impurity="net doping"
mat.occno=<EXPR>	: mat.occno=1
junc.occno=<EXPR>	: junc.occno=1
region.occno=<EXPR>	: region.occno=1
interface.occno=<EXPR>	: interface.occno=1
val.occno=<EXPR>	: val.occno=1
datafile=<QSTRING>	: datafile="results.final"
outfile=<QSTRING>	: outfile="extract.dat"


```

bias.step=<EXPR>      : bias.step=0.25

bias.start=<EXPR>     : bias.start=0.0

bias.stop=<EXPR>      : bias.stop=5.0

temp.val=<EXPR>       : temp.val=300.0

soi                  : FALSE

semi.poly            : FALSE

incomplete           : FALSE

x.val=<EXPR> | y.val=<EXPR> | region=<QSTRING> : x.val is set to be 5% from
                                                left-hand side of structure.

```

5.3.3: Examples of Process Extraction

Note: You can enter `extract` commands on multiple lines using a backslash character for continuation. The syntax, however, shown below should be entered on a single line although shown on two or more lines.

The following examples assume to be extracting values from the current simulation running under DECKBUILD. You can use saved standard structure files directly with `extract` using the syntax below.

```
extract init infile="filename"
```

Material Thickness

Extract the thickness of the top (first) occurrence of Silicon Oxide for a 1D cutline taken where $Y=0.1$ (Assume 2D structure). A warning is then displayed if results cross boundaries set by `max.v` and `min.v`.

```
extract name="tox" thickness material="SiO~2" mat.occno=1 y.val=0.1
min.v=100 max.v=500
```

oxide can be substituted for the material="SiO~2".

Junction Depth

Extract the junction depth of the first junction occurrence in the top (first) occurrence of silicon for a 1D cutline taken where $X=0.1$.

```
extract name="j1 depth" xj material="Silicon" mat.occno=1 x.val=0.1
junc.occno=1
```

Surface Concentration

Extract the surface concentration net doping for the top (first) occurrence of silicon for a 1D cutline taken for an X value corresponding to the gate contact/region for loaded MASKVIEWS cutline data.

```
extract name="surface conc" surf.conc impurity="Net Doping"
material="Silicon" mat.occno=1 region="gate"
```

QUICKMOS 1D Vt

Extract the 1D threshold voltage of a p-type MOS cross section at $x=0.1$ using the built-in QUICKMOS 1D device simulator. This example uses a default gate bias setting of 0-5V for a 0.25V step with the substrate at 0V and a default device temperature of 300 Kelvin. Values of QSS and gate workfunction have also be specified.

```
extract name="1D Vt" ldvt ptype qss=1e10 workfunc=5.09 x.val=0.1
```

This 1D Vt extraction will calculate the 1D threshold voltage of an n-type MOS cross section at $X=0.1$, where a gate voltage range (0.5-20V) was specified while the substrate (V_b) is set at 0.2V. The device temperature has been set to 350 Kelvin.

```
extract name="1D Vt 0-20v" ldvt ntype bias=0.5 bias.step=0.25 bias.stop=20.0  
vb=0.2 temp.val=350.0 x.val=0.1
```

Sheet Resistance and Sheet Conductance

Note: For sheet conductance extraction substitute "sheet.res" with "conduct" (e.g., conduct, p.conduct, n.conduct).

Extract the total sheet resistance of the first p-n region in the top (first) occurrence of polysilicon for a cutline at $X=0.1$. Polysilicon is treated as a metal by default but is flagged here as a semiconductor (semi.poly). The default device temperature of 300 Kelvin and no layer biases will be used and the incomplete ionization flag is also set for carrier freezeout calculations (see "Incomplete Ionization Of Impurities" physics section within the ATLAS manual).

```
extract name="Total SR" sheet.res material="Polysilicon" mat.occno=1  
x.val=0.1 region.occno=1 semi.poly incomplete
```

Extract the n-type sheet resistance of the second p-n region in the top (first) occurrence of silicon for a cutline at $X=0.1$, where the second region is held at 4.0V and the device temperature is set to 325 Kelvin. These commands use the start/cont/done syntax to create a multi-line statement as described in Section 5.8: "Extract Features".

```
extract start material="Silicon" mat.occno=1 region.occno=2 bias=4.0  
x.val=0.1 extract done name="N-type SR" n.sheet.res material="Silicon"  
mat.occno=1 temp.val=325 x.val=0.1 region.occno=2
```

The following multi-line statement extracts the p-type sheet resistance of the first p-n region in the top (first) occurrence of silicon for a cutline at $x=0.1$, where the first region is held at 5.0V. The second region is held at 1.0V and the first interface Qss value equal to 1e10.

```
extract start material="Silicon" mat.occno=1 region.occno=1 bias=5.0  
x.val=0.1  
extract cont material="Silicon" mat.occno=1 region.occno=2 bias=1.0  
x.val=0.1  
extract cont interface.occno=1 qss=1.0e10  
extract done name="P-type SR" p.sheet.res material="Silicon" mat.occno=1  
x.val=0.1 region.occno=1
```

Note: This is an example of the multi-line "start/continue/done" type of statement used to specify layer biases and Qss values. It is recommended that you always let the Extract popup write this particular syntax. The Qss value also specifies the material interface occurrence involved, counting from the top down. There can be any number of additional "continue" lines to specify the biases on other layers and the Qss values of other interfaces; the last line, "done", does the actual extraction.

1D Max/Min Concentration

Extract the peak concentration of net doping within the first p-n region of the top (first) layer of silicon for a 1D cutline at $x=0.1$.

```
extract name="Max 1d Net conc" max.conc impurity="Net Doping"  
material="Silicon" mat.occno=1 x.val=0.1 region.occno=1
```

Extract the peak concentration of phosphorus within any p-n regions (default) for all materials using a 1D cutline at $x=0.1$.

```
extract name="Max 1d phos conc" max.conc impurity="Phosphorus"  
material="All" x.val=0.1
```

Extract the minimum concentration of boron within any p-n regions of the top (first) layer of silicon for a 1D cutline at $x=0.1$.

```
extract name="Min 1d bor conc" min.conc impurity="Boron" material="Silicon"  
mat.occno=1 x.val=0.1
```

2D Max/Min Concentration

Extract the peak concentration of net doping for the entire 2D structure.

```
extract name="Max 2D net conc" 2d.max.conc impurity="Net Doping"  
material="All"
```

Extract the peak concentration of boron within the silicon material in the 2D “box” limits defined.

```
extract name="Max 2D bor conc" 2d.max.conc impurity="Boron"  
material="Silicon" y.min=0.1 y.max=0.9 x.min=0.2 x.max=0.6
```

In addition to this statement, you can add the `interpolate` flag. When present, this flag causes the extraction to perform interpolation at the edges of the specified bounding box for min/max concentration and position.

Extract the minimum concentration of phosphorus for all materials within the 2D “box” limits. These limits are defined by user-defined y coordinates and x values corresponding to loaded MASKVIEWS cutline information for the specified electrode or region.

```
extract name="Min 2D phos conc" 2d.min.conc impurity="Phosphorus"  
material="All" region="gate" y.min=0.1 y.max=0.9
```

The following multi-line extract command measures the minimum concentration of antimony for the entire 2D structure and return the x - y coordinates of the extracted concentration.

```
extract name="Min 2D ant conc" 2d.min.conc impurity="Antimony"  
material="All"  
extract name="Min 2D ant conc X position" x.pos  
extract name="Min 2D ant conc Y position" y.pos
```

Note: The x - y position syntax must directly follow the 2D concentration extraction (same as `start/continue/done` syntax). We advise you to use the Extract popup to create these statements.

2D Concentration File

The output file contains data of the format `x y c`, where `c` is the value of concentration at the coordinates `xy`. The following example extracts the boron concentration in Silicon for the whole structure.

```
extract 2d.conc.file material="silicon" impurity="boron" outfile="conc.dat"
```

1D Material Region Boundary

Extracting the maximum Y boundary (upper side) location of the first occurrence of silicon material for a 1d cutline taken at X=2.

```
extract name="max_y" max.bound material="silicon" x.val=2 mat.occno=1
```

Extracting the minimum X boundary (left side) location of the second occurrence of polysilicon material for a 1d cutline at Y=3.

```
extract name="min_x" min.bound material="polysilicon" y.val=3 mat.occno=2
```

2D Material Region Boundary

Extracting the minimum X boundary (left side) location of the photoresist material region at XY coordinates (7.6, -1.2).

```
extract name="minx" min.bound x.pos material="photoresist" x.val=7.6  
y.val=-1.2
```

Extracting the maximum Y boundary (upper side) location of the photoresist material region at XY coordinates (5.2, 0).

```
extract name="maxy" max.bound y.pos material="photoresist" x.val=5.2 y.val=0
```

2D Concentration Area

Integrates the Boron concentration within the specified “box” limits, using a cutline step of 0.05 microns.

```
extract name="limit_area" 2d.area impurity="Boron" x.step=0.05 x.min=0.01  
y.min=0.23 x.max=0.6 y.max=0.45
```

In addition to this statement, you can add the `interpolate` flag. When present, this flag causes the extraction to perform interpolation at the edges of the specified bounding box for min/max concentration and position.

Integrates the Phosphorus concentration for the whole 2D structure using a cutline step of 0.03 microns.

```
extract name="device_area" 2d.area impurity="Phosphorus" x.step=0.03
```

Note: The `x.step` refers to the number of 1d cutlines used to obtain the 2d area. For a device with an X axis of 7 microns, an `x.step` of 1 would result in 8 cutlines being used at 1 micron intervals.

2D Maximum Concentration File

Creates a Data format file plotting the position of the maximum potential, in silicon material only, for the whole 2D structure. A maximum potential Y position is found for every X step of 0.1 microns. These Data format files can be loaded into TONYPLOT (`-ccd`) to represent a line of maximum concentration across a device.

```
extract name="Total_max_pot" max.conc.file impurity="potential" x.step=0.1  
material="silicon" outfile="totalconc.dat"
```

Creates a Data format file plotting the position of the maximum potential, in any material, for the specified “box” limits. A maximum potential Y position is found for every X step of 0.2 microns.

```
extract name="limit_max_pot" max.conc.file impurity="potential"
x.step=0.2 outfile="limitconc.dat" x.min=0 x.max=7 y.min=0 y.max=0.09
```

Note: The x.step does not refer to cutlines but to the number of X coordinates used. A value of 1 representing stepping 1 micron in X for every max Y value calculated.

QUICKMOS CV Curve

Extract a MOS CV curve, ramping the gate from 0 to 5 volts, with 0 volts on the backside and the device temperature set at 325 Kelvin (default 300 K). This example creates a curve that is stored in file cv.dat and can be shown using TONYPLOT. To bring up TONYPLOT on this file, an easy way is to highlight the file name and then click on DECKBUILD’s **Tools** button. TONYPLOT starts and loads with the file automatically.

```
extract name="CV curve" curve(bias,ldcapacitance vg=0.0 vb=0.0 bias.ramp=vg
bias.step=0.25 bias.stop=5.0 x.val=0.1 temp.val=325) outfile="cv.dat"
```

To get the maximum capacitance for the same curve, insert the keyword max (by editing the syntax created by the popup). Notice that in this example, a single value is being extracted from a curve, not the curve itself. You still, however, store the curve used during the calculation into an output file, which is always the case.

```
extract name="CV curve Max cap" max(curve(bias,ldcapacitance vg=0.0 vb=0.0
bias.ramp=vg bias.step=0.25 bias.stop=5.0 x.val=0.1 temp.val=325))
outfile="cv.dat"
```

To find what the capacitance was at voltage 4.3 volts, use the following syntax:

```
extract name="MOS capacitance at Vg=4.3" y.val from curve(bias,ldcapacitance
vg=0.0 vb=0.0 bias.ramp=vg bias.step=0.25 bias.stop=5.0 x.val=0.1
temp.val=325) where x.val = 4.3
```

The general form of this syntax is

```
extract y.val from curve(xaxis, yaxis) where x.val=number_on_xaxis
```

and:

```
extract x.val from curve(xaxis, yaxis) where y.val=number_on_yaxis
```

where xaxis and yaxis will determine the actual curve. The syntax for this example was created by using the popup to write the syntax for the CV curve, and then adding the y.val... where x.val syntax in the input deck.

For more examples on how to manipulate curves, see the examples in Section 5.4: “Device Extraction”.

Junction Capacitance Curve

Extract a curve of junction capacitance against bias where the first region in the top (first) layer of silicon is ramped from 0 to 5V. Capacitance of the first junction occurrence (upper) is measured and the resultant curve is output to the file XjV.dat. Device temperature is default (300 Kelvin). If only one junction exists for the selected region, you must use then a junction occurrence of one (upper).

```
extract start material="Silicon" mat.occno=1 bias=0.0 bias.step=0.25
bias.stop=5.0 x.val=0.1 region.occno=1
extract done name="Junc cap vs bias" curve(bias,ldjunc.cap
material="Silicon" mat.occno=1 x.val=0.1 region.occno=1 junc.occno=1)
outfile="XjV.dat"
```

Extract the minimum junction capacitance on the created junction capacitance against bias curve. The second region in the top (first) layer of silicon is ramped from 0 to 3V and the capacitance of the second junction occurrence (lower) is measured. Device temperature is set for calculations to be 325 Kelvin. The resultant curve is output to the file XjVmin.dat, while the extracted minimum value is logged to the default results file (results.final).

```
extract start material="Silicon" mat.occno=1 bias=0.0 bias.step=0.25
bias.stop=3.0 x.val=0.1 region.occno=2
extract done name="Junc cap vs bias" min(curve(bias,ldjunc.cap
material="Silicon" mat.occno=1 x.val=0.1 region.occno=2 junc.occno=2
temp.val=325)) outfile="XjVmin.dat"
```

Note: The junction occurrence is only valid for the specified region. In other words, there is only a maximum of two possible junctions for the specified region.

Junction Breakdown Curve

Extract a curve of electron ionization integral against bias where the first region in the top (first) layer of silicon is ramped from 0 to 5V and device temperature is set to be 325 Kelvin. The resultant breakdown curve is output to the file Nbreakdown.dat. See the “Impact” command section and “Impact Ionization” physics section in the ATLAS USER’S MANUAL for the Selberherr model used in calculation.

```
extract start material="Silicon" mat.occno=1 bias=0.0 bias.step=0.25
bias.stop=5.0 x.val=0.1 region.occno=1
extract done name="N Breakdown "curve(bias,n.ion material="Silicon"
mat.occno=1 x.val=0.1 region.occno=1 temp.val=325)
outfile="Nbreakdown.dat"
```

The following extraction creates a curve of hole ionization integral against bias, and calculates the breakdown voltage corresponding to the point where the hole ionization integral intercepts 1.0. The second region in the top (first) layer of silicon is ramped from 0 to 20V and the device temperature is set to the default of 300 Kelvin. The resultant breakdown curve is output to the file Pbreakdown.dat and the breakdown voltage is appended to the default results file (results.final).

```
extract start material="Silicon" mat.occno=1 bias=0.0 bias.step=0.50
bias.stop=20.0 x.val=0.1 region.occno=2
extract done name="P intercept" x.val from curve(bias,p.ion
material="Silicon" mat.occno=1 x.val=0.1 region.occno=2) where y.val=1.0
outfile="Pbreakdown.dat"
```

You can modify the selberherr model parameters using the syntax below. For more information, see Appendix A: “Models and Algorithms”.

```
extract start material="Silicon" mat.occno=1 bias=0.2 bias.step=0.08
bias.stop=5.0 x.val=0.3 region.occ=2
extract done name="iiP" curve(bias, p.ion material="Silicon" mat.occno=1
x.val=0.3 region.occno=2 egran=4.0e5 betap=1.0 betan=1.0
an1=7.03e5 an2=7.03e5 bn1=1.231e6 bn2=1.231e6 ap1=6.71e5 ap2=1.582e6
bp1=1.693e6 bp2=1.693e6) outfile="extract.dat"
```

SIMS Curve

Extract the concentration profile of net doping in the top (first) layer of silicon. The output curve is placed into the file SIMS.dat.

```
extract name="SIMS" curve(depth,impurity="Net Doping" material="Silicon"
mat.occno=1 x.val=0.1) outfile="SIMS.dat"
```

SRP Curve

Extract the **SRP** (Spreading Resistance Profile) in the top (first) silicon layer. The output curve is placed into the file SRP.dat.

```
extract name="SRP" curve(depth, srp materials="Silicon" mat.occno=1
x.val=0.1)
outfile="SRP.dat"
```

The following command will calculate the **SRP** (Spreading Resistance Profile) in the top (first) silicon layer using a specified 100 etch steps of uniform size. The output curve is placed into the file SRP100.dat.

```
extract name="SRP100" curve(depth,srp material="Silicon"
mat.occno=1 n.step=100 x.val=0.5) outfile="srp100.dat"
```

Note: Where `n.step` is not specified, the default is 50 etch steps of variable size dependent on the gradient of net concentration. If `n.steps` is set, uniform etch steps are used.

Sheet Resistance/Conductance Bias Curves

Extract the **Total** sheet conductance against bias curve of the first p-n region in the top (first) occurrence of polysilicon. Polysilicon is treated as a metal by default but is flagged here as a semiconductor (`semi.poly`). The device temperature is set to 325 Kelvin (default=300 Kelvin) and a bias ramped from 0 to 5V on the same polysilicon region.

```
extract start material="Polysilicon" mat.occno=1 bias=0.0 bias.step=0.00
bias.stop=5.0 x.val=0.1 region.occno=1
extract done name="Total SC" curve(bias,ldconduct material="Polysilicon"
mat.occno=1 temp.val=325 x.val=0.1 region.occno=1 semi.poly)
outfile="totalSC.dat"
```

Extract the n-type sheet conductance against bias curve of the first p/n region in the top (first) occurrence of silicon where a bias ramped from 0V to 5V on the same silicon region and a value of QSS (4.0e10) is specified for the first interface occurrence.

```
extract start material="Silicon" mat.occno=1 region.occno=1 bias=0.0
bias.step=0.00 bias.stop=5.0 x.val=0.1
extract cont interface.occno=1 qss=4.0e10
extract done name="N-type SC" curve(bias,ldn.conduct material="Silicon"
mat.occno=1 x.val=0.1 region.occno=1) outfile="NtypeSC.dat"
```

Extract the p-type sheet conductance against bias curve of the first p-n region in the top (first) occurrence of silicon, where a bias ramped from 0 to 5V on the same silicon region and a bias of 2V is held on the first region of the top occurrence of polysilicon. A value of QSS (5.0e10) is also specified for the first interface occurrence.

```
extract start material="Silicon" mat.occno=1 region.occno=1 bias=0.0
bias.step=0.00 bias.stop=5.0 x.val=0.1
extract cont material="Polysilicon" mat.occno=1 bias=2.0 x.val=0.1
region.occno=1
extract cont interface.occno=1 qss=5.0e10
extract done name="P-type SC" curve(bias,ldp.conduct material="Silicon"
mat.occno=1 x.val=0.1 region.occno=1) outfile="PtypeSC.dat"
```

The command below extracts the p-type sheet conductance against bias curve of the first and second p-n regions in the top (first) layer of silicon, where a bias is ramped from 1V to -2V on the top (first) polysilicon layer.

```
extract start material="Polysilicon" mat.occno=1 bias=1.0 bias.step=-0.05
bias.stop=-2.0 x.val=0.01
extract done name="region1+2" curve(bias,1dp.conduct material="Silicon"
mat.occno=1 x.val=0.01 region.occno=1 region.stop=2) outfile="region1+2.dat"
```

Note: For sheet resistance extraction, substitute "1dconduct" with "1dsheet.res" (i.e., 1dsheet.res, 1dnsheet.res, 1dpsheet.res).

Electrical Concentration Curve

Extract the electron distribution against depth for the top (first) layer of silicon where a bias is ramped from 0 to 5V for the first region of the silicon and a QSS of 4.0e10 set for the first interface occurrence. Device temperature is set at 325 Kelvin.

```
extract start material="Silicon" mat.occno=1 region.occno=1 bias=0.0
bias.step=0.00 bias.stop=5.0 x.val=0.1
extract cont interface.occno=1 qss=4.0e10
extract done name="Electrical conc" curve(depth,n.conc material="Silicon"
mat.occno=1 x.val=0.1 temp.val=325) outfile="extract.dat"
```

ED Tree (Optolith)

Create a Data format file plotting a single branch of an ED tree for deviation of 10% from the datum, the specified critical dimension (CD) value of 0.5. The x.step defines the defocus step to be used. 0.08 representing 8% of the total X axis range for each calculation. For each value of defocus at the specified critical dimension deviation, the value of dose is interpolated. Therefore, the resulting curve is dose against defocus for a critical dimension of 0.5 plus 10%.

```
extract name="ed+10" edcurve(da.value."DEFOCUS", da.value."CDs",
da.value."DOSE",dev=10 datum=0.5 x.step=0.08) outf="ed10.dat"
```

Note: If no x.step is specified the actual curve defocus points are used.

Elapsed time

The timer is reset to 10 seconds, a timestamp extracted before and then after a simulation. The elapsed time is then calculated by subtraction.

```
extract name="reset_clock" clock.time start.time = 10
extract name="t1" clock.time
<simulation>
extract name="t2" clock.time
extract name="elapsed_time" $t2 - $t1
```

Note: This extraction does not measure CPU time

5.4: Device Extraction

Device extraction always deals with a “logfile” that contains I-V information produced by a device simulator (such as ATLAS). Therefore, it deals almost exclusively in curves. The following section show how to construct a curve or extract values on a curve for all possible devices. For the special case of MOS devices, both ATLAS and SMINIMOS4 have popups with a number of pre-defined MOS tests. See Section 5.6: “MOS Device Tests” for more information.

Device extraction also deals with structure files, which contain information saved by a device simulator (e.g., ATLAS). You can extract this information by using the process extraction syntax style shown below. The following extracts the total electric field for silicon in a 1-D cutline, where $x = 0.5$ for the loaded device structure file.

```
extract name="test" 2d.max.conc impurity="E Field" material="Silicon"
x.val=0.5
```

There are some differences between the syntax used by EXTRACT and the syntax used by the ATLAS output command. Appendix B: “Extract and ATLAS Syntax” shows these differences.

EXTRACT allows you to construct a curve using separate X and Y axes. For each axis, you can choose the voltage or current on any electrode, the capacitance or conductance between any two electrodes, or the transient time for AC simulations. You can either manipulate the axes individually, such as multiplication or division by a constant, or combine axes in algebraic functions.

Note: The curve manipulation discussed is equally applicable to all curves, whether the curve came from process or device simulation. The only type-specific syntax relates to the curve axes. For example, gate voltage can't be extracted from a process simulator. If you try, then a warning message will appear.

5.4.1: The Curve

The basic element is always the curve. Once the curve is constructed, it can be used as is, by saving it to a file for use by TONYPLOT, or as an OPTIMIZER target, or it can be used as the basis for further extraction. For details on the extract curve syntax, see Section 5.3.1: “Extract Syntax”.

To construct a curve representing voltage on electrode “emitter1” (on the X axis) versus current on electrode “base2”, write:

```
extract name="iv" curve(v."emitter1", i."base2")
```

The first variable specified inside the parentheses becomes the X axis of the curve. The second variable becomes the Y axis. The v. “name” and i. “name” syntax is used for any electrode name — just insert the proper name of the electrode. The electrode name be defined previously (such as in the device deck, or previous to that in an ATHENA input deck using the electrode statement, or interactively in DEVEDIT). Electrode names may contain spaces but must always have quotation marks.

Transient time is represented by the keyword time.

```
extract name="It curve" curve(time, i."anode")
```

For Device temperature curves, use:

```
extract name="VdT" curve(v."drain", temperature)
```

For extracting a frequency curve use:

```
extract name="Idf" curve(i."drain", frequency)
```

To extract a capacitance or conductance curve, use this syntax:

```
extract name="cv" curve(c."electrode1"electrode2", v."electrode3")
```

and

```
extract name="gv" curve(g."electrode1" "electrode2", v."electrode3")
```

For other electrical parameters (see Section 5.3.1: “Extract Syntax” section for valid electrical parameters) use the following syntax:

```
extract name="IdT" curve(elect."parameter", v."drain")
```

An `extract name` is given in each example. Although optional, it is always a good idea to name `extract` statements so they can be identified later. Names are always necessary for entering an `extract` statement in DECKBUILD’s OPTIMIZER, and for recognition by the VWF.

It is also possible to shift or manipulate curve axes. Each axis is manipulated separately. The simplest form of axis manipulation is algebra with a constant.

```
extract name="big iv" curve(v."gate"/50, 10*i."drain")
```

You can multiply, divide, add, or subtract any constant expression to each axis.

Curve axis can also be combined algebraically, similar to TONYPLOT’s function capability:

```
extract name="combine" curve(i."collector", i."collector"/i."base")
```

All electrode values (current, voltage, capacitance, conductance) can be combined in any form this way.

Another curve type is `deriv()` used to return the derivative (`dydx`). For example, statement below will create the curve of `dydx` gate bias and drain current plotted against and X axis of gate bias.

```
extract name="dydx" deriv(v."gate", i."drain")
outfile="dydx.dat"
```

It is also possible to calculate `dydx` to the `nth` derivative as below.

```
extract name="dydx2" deriv(v."gate", i."drain", 2)
outfile="dydx2.dat"
```

To find local maxima and minima on a curve, limit the section of the curve X axis. The following statement extracts the maximum drain current, where gate bias is between the limits of 0.5 volts and 2.5 volts.

```
extract name="limit" max(curve(v."gate", i."drain", x.min=0.5
x.max=2.5)) outf="limit.dat"
```

In addition, there are several operators which apply to curve axes. They are as follows:

```
abs(axis)
log(axis)
log10(axis)
sqrt(axis)
atan(axis)
-axis
```

For instance:

```
extract curve(abs(i."drain"), abs(v."gate"))
```

The operators can be combined. For example, `log10(abs(axis))`. These operators also work on curve axes from process simulation.

5.4.2: Curve Manipulation

A number of curve manipulation primitives exist:

```
min(curve)
max(curve)
ave(curve)
minslope(curve)
maxslope(curve)
slope(line)
xintercept(line)
yintercept(line)
area from curve
area from curve where x.min=X1 and x.max=X2
x.val from curve where y.val=k
y.val from curve where x.val=k
x.val from curve where y.val=k and val.occno=n
y.val from curve where x.val=k and val.occno=n
grad from curve where y.val=k
grad from curve where x.val=k
```

For details on Extract curve manipulation syntax, see Section 5.3.1: “Extract Syntax”.

For instance, using the BJT curve example above, you could find the maximum of I_c/I_b vs I_c , or maximum beta, by writing:

```
extract name="max beta" max(curve(i."collector", i."collector"/i."base"))
```

`max()`, `min()`, and `ave()` all work on the Y axis of the curve.

The sloped lines and intercepts often work together. The primitives `minslope()` and `maxslope()` can be thought of as returning a line. Extracting a line by itself has no meaning, so three other operators take a line as input. The operators are `slope()`, which returns the slope of the line, and `xintercept()` and `yintercept()`, which return the value where the line intercepts the corresponding axis.

For instance, a V_t test for MOS devices looks at a curve of V_g (x) versus I_d (y) and finds the X intercept of the maximum slope. Such a test would look like:

```
extract name="vt" xintercept(maxslope(curve(abs(v."gate", abs(i."drain"))))
```

Some V_t tests take off $V_d/2$ from the resulting value. You could write:

```
extract name="vt" xintercept(maxslope(curve(abs(v."gate",
abs(i."drain")))) - ave(v."drain")/2
```

Note that the last example uses:

```
ave(v."drain")/2
```

The `max()`, `min()`, and `ave()` operators can be used on both curves,

```
extract name="Iave" ave(curve(v."gate", i."drain"))
```

and also on individual curve axes,

```
extract name="Iave" ave(i."drain")
```

or even on axis functions:

```
extract name="Icb max" max(i."collector"/i."base")
```

You can also find the Y value on a curve for a given X value and the other way round. For example, to find the collector current (Y) for base voltage 2.3 (X), use:

```
extract name="Ic[Vb=2.3]" y.val from curve(abs(v."base"),
abs(i."collector")) where x.val = 2.3
```

EXTRACT uses linear interpolation if necessary. If more than one point on the curve matches the condition, EXTRACT takes the first one, unless you use the following syntax to specify the occurrence of the condition. This example would find the second Y point on the curve matching an X value of 2.3.

```
extract name="Ic[Vb=2.3]" y.val from curve(abs(v."base",
abs(i."collector"))
where x.val = 2.3 and val.occno =2
```

The condition used for finding an intercept can be a value or an expression and therefore use the `min()`, `max()`, and `ave()` operators. The following command creates a transient time against drain-gate capacitance curve and calculates the intercepting time where the capacitance is at its minimum value.

```
extract name="t at Cdrain-gate[Min]" x.val from curve(time, c."drain"gate")
where y.val=min(c."drain"gate")
```

In addition to finding intercept points on curves, you can also calculate the gradient at the intercept, specified by either a Y or X value as shown below.

```
extract name="slope_at_x" grad from curve(v."gate", i."drain")
where x.val=1.5
extract name="slope_at_y" grad from curve(v."gate", i."drain")
where y.val=0.001
```

You can also find the area under a specified curve for either the whole curve or as below between X limits.

```
extract name="iv area" area from curve(v."gate", c."drain"gate")
where x.min=2 and x.max=5
```

5.4.3: BJT Example

As a final example for device extraction, consider finding, say, the beta value for a BJT device, at 1/10th the current for max beta. This example sums up the information presented so far, and also introduces the feature of variable substitution.

First, you need to figure out what the current is at max beta. Max beta was presented in a previous example:

```
extract name="max beta" max(curve(i."collector", i."collector"/i."base"))
```

After this statement has been run, extract remembers the extract name, max beta, and the resulting value. Use this information later on using variable substitution. In this example, you need to get the current, or X axis value, at max beta, to figure out what 1/10th of it is. To do this, use the extracted max beta as our Y axis "target value":

```
extract name="Ic[max beta]" x.val from curve(i."collector",
i."collector"/i."base") where y.val=$"max beta"
```

Finally, extract the value of I_c/I_b for $I_c = \text{max beta}/10$.

```
extract name="Ic/Ib for Ic=Ic[max beta]/10" y.val from curve(i."collector",
i."collector"/i."base") where x.val=$"Ic[max beta]/10"
```

For more information about variable substitution, see Section 5.8: "Extract Features".

5.5: General Curve Examples

The following examples assume that they are extracting values from the currently loaded logfile running under DECKBUILD. Saved “IV” log files, however, can be used directly with `extract` using the syntax below.

```
extract init infile="filename"
```

Note: You can enter `extract` commands on multiple lines using a backslash character for continuation. You should, however, enter the syntax shown below on a single line although shown on two or more lines.

5.5.1: Curve Creation

The following command extracts a curve of collector current against base voltage and places the output in `icvb.dat`.

```
extract name="IcVb curve" curve(i."collector", v."base") outfile="icvb.dat"
```

5.5.2: Min Operator with Curves

The following command calculates the minimum value for a curve of drain current against internal gate voltage.

```
extract name="Vgint[Min]" min(curve(i."drain", vint."gate"))
```

5.5.3: Max Operator with Curves

The following command calculates the maximum value for a curve of base voltage against base-collector capacitance.

```
extract name="Cbase-coll[Max]" max(curve(v."base", c."base" "collector"))
```

5.5.4: Ave Operator with Curves

The following command calculates the average value for a curve of drain current against gate-drain conductance.

```
extract name="Ggate-drain[Ave]" ave(curve(i."drain", g."gate" "drain"))
```

5.5.5: X Value Intercept for Specified Y

The following command creates a frequency against drain current curve and calculates the intercepting frequency for a drain current of $1.5\text{-e}6$.

```
extract name="Freq at Id=1.5e-6" x.val from curve(frequency, i."drain") where  
y.val=1.5e-6
```

5.5.6: Y Value Intercept for Specified X

The following command creates a drain voltage against device temperature curve and calculates the intercepting temperature for a drain voltage of 5V.

```
extract name="T at Vd=5" y.val from curve(v."drain", temperature) where  
x.val=5.0
```

5.5.7: Abs Operator with Axis

The following command creates a curve of absolute gate voltage against absolute optical wavelength (log, log10 and sqrt also available).

```
extract name="Vg-optW curve" curve(abs(v."gate"), abs(elect."optical wavelength"))
```

5.5.8: Min Operator with Axis Intercept

The following command creates a transient time against gate-drain capacitance curve and calculates the intercepting time where the capacitance is at its minimum value.

```
extract name="t at Cgate-drain[Min]" x.val from curve(time, c."gate" "drain")
where y.val=min(c."gate" "drain")
```

5.5.9: Max Operator with Axis Intercept

The following command creates a collector current against collector current divided by base current curve and calculates the intercepting collector current where I_c/I_b is at a maximum value.

```
extract name="Ic at Ic/Ib[Max]" x.val from curve(i."collector",
i."collector"/i."base") where y.val=max(i."collector"/i."base")
```

5.5.10: Second Intercept Occurrence

The following command creates a gate voltage against source photo current curve and calculates the second intercept of gate voltage for a source photo current of $2e-4$.

```
extract name="2nd Vg at Isp=2e-4" x.val from curve(v."gate", elect."source
photo current") where y.val=2e-4 and val.occno=2
```

5.5.11: Gradient at Axis Intercept

The following command creates a probe Itime against drain current curve and finds the gradient at the point where probe Itime is at a maximum.

```
extract name="grad_at_maxTime" grad from curve(probe."Itime",
i."drain") where y.val=max(probe."Itime")
```

5.5.12: Axis Manipulation with Constants

The following command creates a gate voltage divided by ten against total gate capacitance multiplied by five. Adding and subtracting are also available.

```
extract name="Vg/10 5*C-gg curve" curve(v."gate"/10, 5*c."gate" "gate")
```

5.5.13: X Axis Interception of Line Created by Maxslope Operator

The following command calculates the X axis intercept for the maximum slope of a drain current against gate voltage curve.

```
extract name="Xint for IdVg" xintercept(maxslope(curve(i."drain",
v."gate")))
```

5.5.14: Y Axis Interception of Line Created by Minslope Operator

The following command calculates the Y axis intercept for the minimum slope of a substrate current against drain voltage.

```
extract name="Yint for IsVd" yintercept(minslope(curve(i."substrate",
v."drain")))
```

5.5.15: Axis Manipulation Combined with Max and Abs Operators

The following command calculates the maximum value of drain-gate resistance.

```
extract name="Rdrain-gate[Max]" max(1.0/(abs(g."drain" "gate")))
```

5.5.16: Axis Manipulation Combined with Y Value Intercept

The following command creates a gate voltage against drain-gate resistance and calculates the intercepting drain-gate resistance for a gate voltage of 0V.

```
extract name="Rdrain-gate at Vg=0" y.val from curve (v."gate", 1.0/
abs(g."drain" "gate"))
where x.val=0.0
```

5.5.17: Derivative

The following command creates the curve of dydx gate bias and drain current plotted against and X axis of gate bias.

```
extract name="dydx" deriv(v."gate", i."drain")
outfile="dydx.dat"
```

This further example calculates to the 2nd derivative.

```
extract name="dydx2" deriv(v."gate", i."drain", 2)
outfile="dydx2.dat"
```

5.5.18: Data Format File Extract with X Limits

The following command finds the local maximum in Data Format file for the curve of v_{in} between 2 and 5 volts against power.

```
extract name="max[2-5]" max(curve(da.value."vin", da.value."power",
x.min=2 x.max=5)) outf="max2-5.dat"
```

5.5.19: Impurity Transform against Depth

The following command calculates the electron concentration in the first occurrence of silicon material for a cutline of $X=1$ squared against depth.

```
{fixed} extract name="nconc^2" curve(depth, (n.conc material="Silicon"
mat.ocno=1 x.val=1) * (n.conc material="Silicon" mat.ocno=1 x.val=1))
outfile="nconc.dat"
```

5.6: MOS Device Tests

A list of ready-made MOS extract statements is also provided. Use them directly or make modifications to suit testing needs. DECKBUILD allows you to create, modify, and save tests.

The following MOS tests are:

- Vt
- Beta
- Theta
- Leakage
- Bvds
- Idsmax
- SubVt
- Isubmax
- Vg[Isubmax]

Do the following to access the list of MOS extract routines.

- **ATLAS:** Choose **Commands**→**Extracts**→**Device...** and the ATLAS Extraction popup will appear. Choose the desired test and click on the **WRITE** button to insert the test into the input deck. Using the **User defined** option, you can enter custom extracts into the popup and save them as defaults.

When you click the **Write Deck** button on the **Control** popup, the extract syntax will be written automatically to the deck along with the selected tests (Figure 5-5).

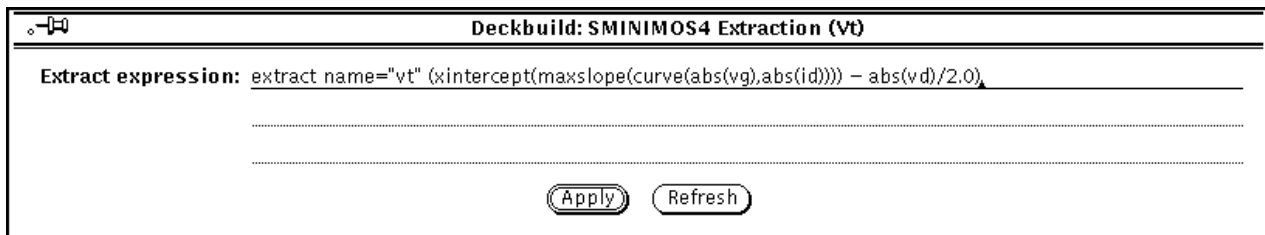


Figure 5-5: The ATLAS Extraction (Vt) Popup

5.7: Extracted Results

Extracted results appear both with the simulator output in the **tty** subwindow and in a special file named by default `results.final`. You can name the file using the `datafile="filename"` syntax. Use the file to compare the results from a large number of runs. For example, if using DECKBUILD's built-in optimizer, the file gives a concise listing of all the results as a function of the input parameters. The extract results file is created in the current working directory.

5.7.1: Units

- Material thickness (angstroms)
- Junction depth (microns)
- Impurity concentrations (impurity units, typically atoms/cm³)
- Junction capacitance (Farads/cm²)
- QUICKMOS capacitance (Farads/cm²)
- QUICKMOS 1D V_t (Volts)
- QUICKBIP 1D solver (see the QUICKBIP section)
- Sheet resistance (Ohm/square)
- Sheet conductance (square/Ohm)
- Electrode voltage (Volts)
- Electrode internal voltage (Volts)
- Electrode current (Amps)
- Capacitance (Farads/micron)
- Conductance (1/Ohms)
- Transient time (Seconds)
- Frequency (Hertz)
- Temperature (Kelvin)
- Luminescent power (Watts/micron)
- Luminescent wavelength (Microns)
- Available photo current (Amps/micron)
- Source photo current (Amps/micron)
- Optical wavelength (Microns)
- Optical source frequency (Hertz)
- Current gain (dB)
- Unilateral power gain frequency (dB)
- Max transducer power gain (dB)

If desired, you can perform whatever unit shifting required by adding the appropriate constants in the device extract tests and saving them as the default. The units are always printed out along with the extract results for built-in single value extract routines. Custom extract routines do not show units.

5.8: Extract Features

5.8.1: Extract Name

`extract` statements should almost always be given names. The name must be prepended to the remainder of the `extract` statement. For example:

```
extract name="gateox thickness" oxide thickness x.val=1.0
```

The `extract` name is used in three ways. The name appears on the OPTIMIZER worksheet when you enter the `extract` statement as a target, and on the VWF worksheet as an extracted parameter. It can also be used in further `extract` statements to perform variable substitution. The name can contain spaces.

5.8.2: Variable Substitution

The `extract` parser maintains a list of variables, each of which consists of a name and a value. A name is defined by any previous named `extract` statement. The corresponding value is the result of the statement.

To refer to a variable's value, precede it with a '\$'. Quotes are optional around variable references, except when the variable name contains spaces, in which case the \$ must precede the quotes. The substituted variable acts as a floating point number, and can be used in any `extract` expression that uses numerical arguments.

For example:

```
extract name="xj1" xj silicon junc.occno=1
extract name="xj2" xj silicon junc.occno=2
extract name="deltaXj" abs($xj1 - $xj2)
```

Examples with spaces:

```
extract name="max boron" max.conc boron
extract name="max arsenic" max.conc arsenic
extract name="PN ratio" $"max boron"/$"max arsenic"
```

You can also use variable substitution in `extract` with the `set` command as shown below.

```
set cutline=0.5
extract name="gateox thickness" oxide thickness x.val=$cutline
```

In addition, filenames to be loaded can also be specified this way. For example:

```
set efile = structure.str
extract init infile="$'efile'"
```

Note: Single quotes can be used to substitute where \$-variable must appear within double quotes.

5.8.3: Min and Max Cutoff Values

Statements may contain `min.val=value` or `max.val=value` or both to define a valid range for extracted results (single-valued results only, not curves). If you do not define either `max` or `min`, then the range extends from +infinity to the stated value respectively. If the extracted value is outside the range, then an error message is printed along with the extracted results and also appended to the default results file.

5.8.4: Multi-Line Extract Statements

EXTRACT statements may be spread over multiple lines to specify layer biases and QSS values as shown in above examples. This involves using the `start/cont/done` syntax.

5.8.5: Extraction and the Database (VWF)

When run with the VIRTUAL WAFER FAB, all extract values in the deck appear as output result columns on the split worksheet. Each row of the worksheet contains the input parameters used to create the results. The extracted value cell values are filled in automatically as the split points complete. If some extracts are only intermediate calculations and are not required to be included in the results worksheet the hide flag can be used. This prevents unrequired extract results from cluttering the worksheet data.

The min/max extract ranges, if defined, are examined. If any extracted value is out of range, then children of that deck fragment (any part of the worksheet that uses the simulation results of that deck fragment) are automatically de-queued and marked with a parent error. The fragment is marked with a range error. The purpose here is that the system does not waste its time by running any simulation beyond that point in the input deck where the range error occurred, for all parts of the split tree that use the particular values of the deck.

5.9: QUICKBIP Bipolar Extract

QUICKBIP is a 1D simulator for bipolar junction transistors (BJT) and is fully integrated inside the DECKBUILD environment. It is accessed using the `extract` command and is available for use with any Silvaco simulator.

The doping profile passed to the QUICKBIP solver should be a bipolar profile. At least, three regions must exist. The top region in the first silicon layer is taken to be the emitter. There may be other materials on top of the silicon.

QUICKBIP can be used with either ATHENA (2-D process simulation) or SSUPREM3 (1-D process simulation). It is used in cases where a 1-D device simulation is both easier and faster to turn around a result. Examples of the QUICKBIP `extract` command language are listed as follows:

```
extract name="bip test bf" bf
extract name="bip test nf" nf
extract name="bip test is" gpis
extract name="bip test ne" ne
extract name="bip test ise" ise
extract name="bip test cje" cje
extract name="bip test vje" vje
extract name="bip test mje" mje
extract name="bip test rb" rb
extract name="bip test rbm" rbm
extract name="bip test irb" irb
extract name="bip test tf" tf
extract name="bip test cjc" cjc
extract name="bip test vjc" vjc
extract name="bip test mjc" mjc
extract name="bip test ikf" ikf
extract name="bip test ikr" ikr
extract name="bip test nr" nr
extract name="bip test br" br
extract name="bip test isc" isc
extract name="bip test nc" nc
extract name="bip test tr" tr
```

Any name can be assigned to each command. In the case of a 2-D simulator, the lateral position of the vertical profile has to be specified with the parameter `x.val=n`. For example:

```
extract name = "forward transit time" tf x.val=0.3
```

Alternatively, a boolean region can be specified when running in conjunction with the IC Layout interface. For example:

```
extract name="my test" tf region="pnp_active_poly"
```

In this case, the bipolar test is performed only in the case where an IC layout cross section intersects the named region.

You can modify QUICKBIP tuning parameters for using the syntax shown below. Appendix A: “Models and Algorithms” provides a more detailed explanation.

```
extract name="Tuning bf" bf x.val=0.5 bip.tn0=1.0e-5 bip.tp0=1.0e-3
bip.an0=2.9e-31 bip.ap0=0.98e-31 bip.nsrhn=5.0e12 bip.nsrhp=5.0e15
bip.betan=2.1 bip.betap=1.
```

Table 5-1 shows the `extract` parameters representing the BJT parameters.

Table 5-1: BJT Parameters

Parameter	Description	Units
bf	Ideal Maximum Forward Beta	
nf	Forward current Emission Coefficient	
gpis	Transport saturation current (IS)	A/cm2
ne	Base-Emitter Leakage Emission Coefficient	
ise	Base-Emitter Leakage Saturation Current	A/cm2
cje	Base-Emitter Zero Bias DEpletion Capacitance	F/cm2
vje	Base-Emitter built in potential	V
mje	Base-Emitter exponential factor	
rb	Zero bias base resistance	Ohms/square
rbm	Minimum base resistance at high current	Ohms/square
irb	Current at half base resistance value	A/cm2
tf	Ideal forward transit time (1/ft)	secs
cjc	Base-Collect zero bias depletion capacitance	F/cm2
vjc	Base-Collector built in potential	V
mjc	Base-Collector exponential factor	
ikf	Corner of Forward Beta High current roll-off	A/cm2
ikr	Corner of Reverse Beta High current roll-off	A/cm2
nr	Reverse Current Emission Coefficient	
br	Ideal Maximum Reverse Beta	
isc	Base-Collector Leakage Saturation Current	A/cm2
nc	Base-Emitter Leakage Emission Coefficient	
tr	Ideal forward transit time	secs

Automated command writing is accomplished with the use of the **DeckBuild Extract** popup window. This is accessed from the **Commands** menu when either SSUPREM3 or ATHENA is selected as the current simulator.

I-V Curves can be visualized with TONYPLOT if the **Compute I-V curve** option is selected on the EXTRACT popup. In this case, select from either forward or reverse characteristics and specify the axes of the curve.

- All extracted parameters can be used as optimization targets.
- All extracted parameters are appended to the default results file in the current working directory. Unless specified, using the `datafile=filename` syntax, it defaults to `results.final`.
- When running under the VWF, all extracted parameters will be logged for regression modeling.

QUICKBIP solves fundamental system of semiconductor equations, continuity equations for electrons and holes, and Poisson's equation for potential self-consistently using the Gummel method. The following physical models are taken into account by QUICKBIP:

- Doping-dependent mobility
- Electric field dependent mobility
- Band gap narrowing
- Shockley-Read-Hall recombination
- Auger recombination

QUICKBIP is fully automatic so that it is unnecessary to specify input biases. QUICKBIP calculates both forward and inverse characteristics of the BJT. For an n-p-n device, these sets are as follows:

1. $V_{eb} = -0.3 \dots -V_{eb_final}$, $V_{eb_step} = -0.025$, $V_{cb} = 0$ V
2. $V_{cb} = -0.3 \dots -V_{cb_final}$, $V_{cb_step} = -0.025$, $V_{eb} = 0$ V
3. V_{eb_final} and V_{cb_final} depend on the particular BJT structures, usually about $-1 \dots -1.5$ (high injection level).

For a p-n-p device, all signs are changed.

5.10: Using Extract with ATLAS

Do the following to calculate extract parameters during an ATLAS simulation.

1. Include an output statement in your original input deck that specifies the parameters of interest (e.g., output charge to specify charge concentration). You cannot extract a parameter unless you specify that parameter (either explicitly or by default) in an output statement.
2. Insert an extract statement to extract the desired parameters (see Chapter 5: “DeckBuild:Extract”).

There are some differences between the EXTRACT syntax and the syntax used by the ATLAS output statement. To extract parameters, use the correct extract statement syntax (not the ATLAS Output statement syntax). For example, the ATLAS Output statement uses E.Field to specify electric field, while the extract statement requires the name E.Field. To extract electric field include the following lines in the input deck:

```
Output E.Field
...
...
...
Extract ... Impurity="E Field"
```

The following table shows the differences between the ATLAS syntax and the extract statement syntax.

ATLAS Parameter	ATLAS Default	Extract Parameter	Units
POTENTIAL	Mandatory	Potential	V
NET DOPING	Mandatory	Net Doping	atoms/cm ³
ELECTRON CONCENTRATION	Mandatory	Electron Conc	cm ³
HOLE CONCENTRATION	Mandatory	Hole Conc	cm ³
CHARGE	False	Change Conc	atoms/cm ³
CON.BAND	False	Conduction Band Energy	V
E.FIELD/EFIELD	False	E Field	V/cm
E.MOBILITY	False	e- Mobility	cm ² /Vs
E.TEMP	True	Electron Temp	K
E.VELOCITY	False	Electron Velocity	cm/s
EX.FIELD	True	E Field X	V/cm
EX.VELOCITY	False	e- Velocity X	m/s
EY.FIELD	False	E Field Y	V/cm
EY.VELOCITY	False	e- Velocity Y	m/s
FLOWLINES	False	Current Flow	None
H.MOBILITY	False	h+ Mobility	cm ² /Vs
H.TEMP	True	Hole Temp	K
H.VELOCITY	False	Hole Velocity	cm/s
HX.VELOCITY	False	h+ Velocity X	m/s

ATLAS Parameter	ATLAS Default	Extract Parameter	Units
HY.VELLOCITY	False	h+ Velocity Y	m/s
IMPACT	True	Impact Gen Rate	scm ³
JY.ELECTRON	False	Je- Y	A/cm ²
J.ELECTRON	True	Je- Current Magnitude	A/cm ²
JX.ELECTRON	FALSE	Je- X	A/cm ²
J.CONDUC	True	Conduction Current	A/cm ²
J.DISP	False	Displacement Current	A/cm ²
J.HOLE	True	h+ Current Magnitude	A/cm ²
J. TOTAL	True	Total Current Density	A/cm ²
JX.CONDUC	False	Cond Current X	A/cm ²
JX.HOLE	False	Jh+ X	A/cm ²
JX. TOTAL	False	Jtot X	A/cm ²
JY.CONDUC	False	Cond Current Y	A/cm ²
JY.HOLE	False	Jh+ Y	A/cm ²
JY. TOTAL	False	Jtot Y	A/cm ²
PHOTOGEN	True	Photo Generation Rate	scm ³
QFN	True	Electron QFL	V
QFP	True	Hole QFL	V
QSS	False	Interface Charge	cm ²
RECOMB	True	Recombination Rate	scm ³
TOT.DOPING	False	Total Doping	atoms/cm ³
TRAPS	True	Traps	cm ³
U.AUGER	False	Auger Recomb Rate	scm ³
R.RADIATIVE	False	Radiative Recomb Rate	scm ³
U.SRH	False	SRH Recomb Rate	scm ³
VAL.BAND	False	Valence Band Energy	V
X.COMB	False	Composition X	None
Y.COMB	True	Composition Y	None
OPT.INTENS	False	Optical Intensity	W/cm ²
OX.CHARGE	False	Fixed Oxide Charge	cm ³

6.1: Overview

The OPTIMIZER is a mechanism that automatically varies one or more input parameters to obtain simulated results, using those parameters which match one or more targets. The OPTIMIZER runs through a number of iterations until the results match the targets within a certain tolerance.

The OPTIMIZER uses the Modified Levenberg-Marquart algorithm to build a response surface of results versus input parameters as the iterations progress. This response surface is used to calculate the input parameter values for each iteration. If for some reason the OPTIMIZER cannot achieve convergence, it stops and display the error condition.

In practice, several parameters are measured and the simulation is then tuned to those values. For example, MOS structure, gate oxide thickness, Vt curves, I-V curves, and a SIMS profile can all be used to tune the input deck with the OPTIMIZER.

The OPTIMIZER is controlled through an easy-to-use graphical worksheet. The worksheet allows you to enter and edit input parameters, targets, and setup information (such as error tolerance) used by the OPTIMIZER. There is also a real-time graphical results display that visualizes input parameter values, target values, and error terms so you can easily track the OPTIMIZER's iterations.

6.1.1: Features

The OPTIMIZER eliminates guesswork by determining the input parameter values necessary to match one or more targets quickly and accurately. Furthermore, since the OPTIMIZER is built on top of DECKBUILD's native auto-interfacing capability, parameters in one simulator can be optimized against the extracted results from a different simulator. For example, optimizing can be against a Vt measurement and an I-V curve simulated in ATLAS using process input parameters in SSUPREM3 or ATHENAor both.

The OPTIMIZER is most useful for tuning studies. Use it to calibrate extracted simulation results to measured data by changing coefficients, such as diffusion and segregation; rather than trying to vary settings, such as simulated time and temperature. You can use such well-tuned input deck fragments, each of which performs a known operation, to build entire input decks. For design studies, rather than tuning studies, we recommend the VIRTUAL WAFER FAB (VWF). The VWF constructs and allows visualization of the entire process response surface, opposed to meeting a single target. It also acts in unison with the OPTIMIZER by storing the well-tuned input deck operations for later use.

Since the input deck requires no modification and no special statements, you can easily optimize any existing deck. When there are satisfying optimization results, the OPTIMIZER can copy the final parameter values back into the deck.

You can save all parameter and target data from the worksheet to disk and reload it at any time.

6.1.2: Terminology

This chapter makes reference to input parameters and targets. Input parameters, or just parameters, are any numerical constants in the input deck. Examples include implant energy, diffusion time, and gate voltage. Targets are values (either single points or entire curves) that are extracted from the simulated results. Examples include oxide thickness, Vt, and a curve of net concentration versus depth.

6.2: Using The Optimizer

6.2.1: Overview

Using the OPTIMIZER is easy. You must perform the following steps once an input deck runs with DECKBUILD.

1. Define the input parameters
2. Define the targets
3. Optionally, define the setup information
4. Start the optimization

6.2.2: The Optimizer Window

The Optimizer is controlled from the **DeckBuild Optimizer** window (Figure 6-1). To display the **Optimizer** window, select **Main Control→Optimizer...** in DECKBUILD.

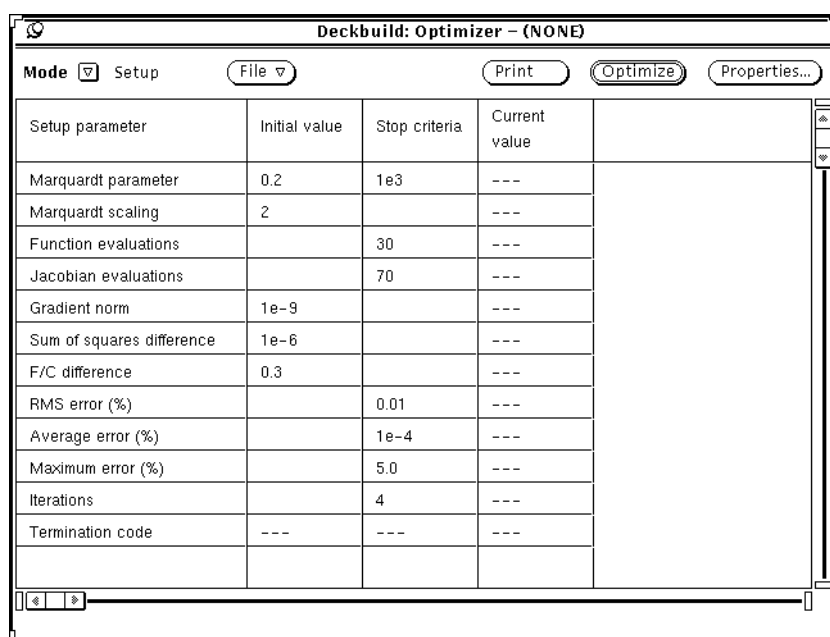


Figure 6-1: Optimizer Window

6.2.3: Optimization Modes

The **Mode** menu is located at the top left of the **Optimizer** window (Figure 6-2). This menu is used to select the five screens displayed in the **Optimizer** window.

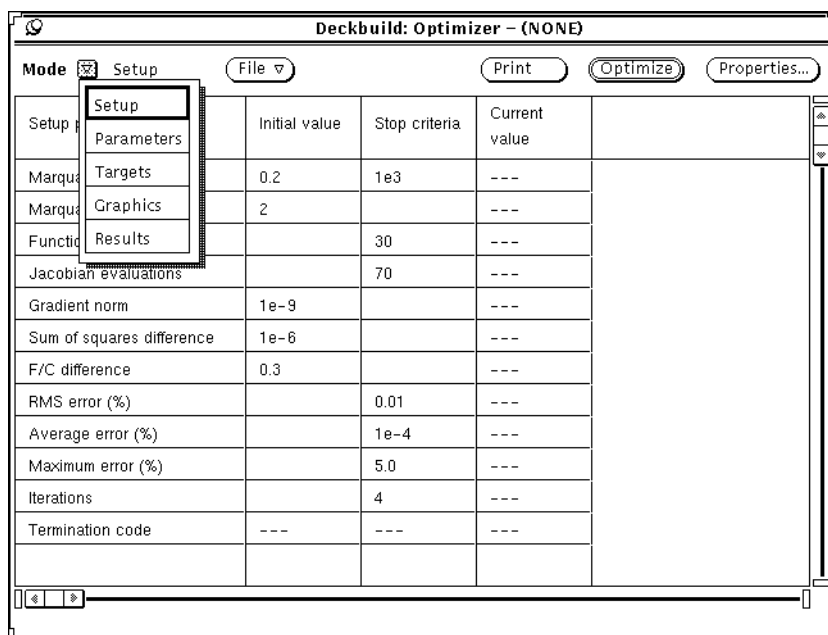


Figure 6-2: Optimizer Modes

The five modes in this window are:

- **Setup** — Optimization setup worksheet.
- **Parameters** — Input parameter definition worksheet.
- **Targets** — Target definition worksheet.
- **Graphics** — Graphical visualization of the optimization run.
- **Results** — Worksheet of parameter and target values for each iteration of a run.

You can define parameters, targets, and setup information for an optimization run in any order. The OPTIMIZER, however, does not run until you define at least one parameter and one target.

6.2.4: Optimizing

Once the parameter, target, and optional setup information is defined, the OPTIMIZER can be started by clicking on the **Optimize** button. Clicking on the button a second time aborts the optimization.

Once the optimization has started, all worksheet information, as well as the input deck, is set to a read-only state that cannot be edited.

During the optimization run, the **Graphics** mode shows real-time updates of current parameter, target, and error values.

At the beginning of a run, the OPTIMIZER always runs a sensitivity analysis of $n+1$ loops at the start of the input deck. n is the number of input parameters.

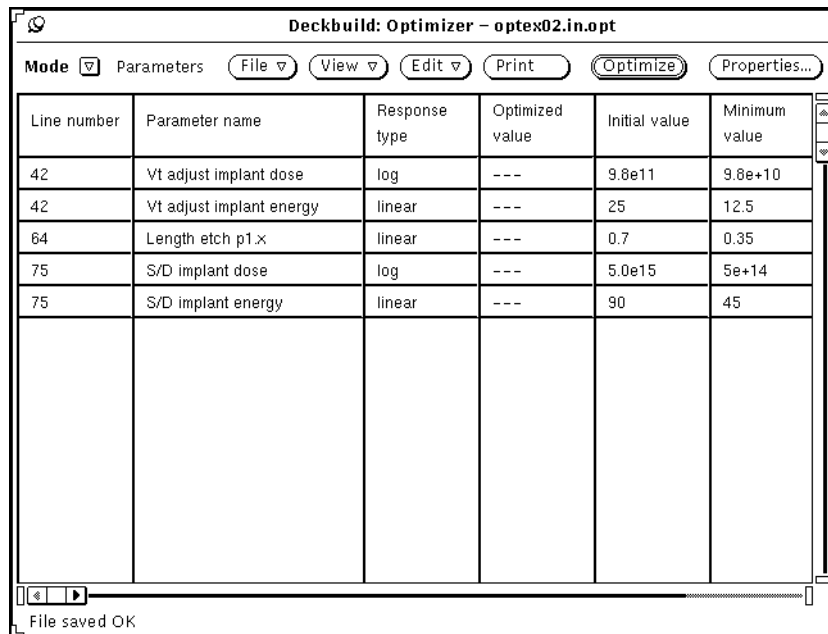
6.3: Parameters

A parameter is any numeric constant in an input deck. That constant may represent anything, such as etch thickness, contact voltage, or work function. Because the OPTIMIZER needs to vary its value to achieve convergence, the parameter must be numeric.

6.3.1: Adding Parameters

To add a parameter, follow these four steps:

1. Display the Parameter worksheet by setting **Mode to Parameters**. To do this, position the pointer over **Mode**, click the MENU mouse button, and select **Parameters**. The **Parameter** worksheet is then displayed (Figure 6-3).



Line number	Parameter name	Response type	Optimized value	Initial value	Minimum value
42	Vt adjust implant dose	log	---	9.8e11	9.8e+10
42	Vt adjust implant energy	linear	---	25	12.5
64	Length etch p1.x	linear	---	0.7	0.35
75	S/D implant dose	log	---	5.0e15	5e+14
75	S/D implant energy	linear	---	90	45

Figure 6-3: Parameter Worksheet with 5 Parameters Defined

2. Select (highlight) the line in the input deck containing the parameter(s) to be added. To do this, position the pointer over the line, and triple-click the SELECT mouse button to capture the entire line. The OPTIMIZER also accepts lines that have only a word or some selected characters.
3. Choose **Add** from the **Edit** pull-down menu. To do this, move the pointer over the **Edit** button, click on MENU and select **Add** (Figure 6-4).

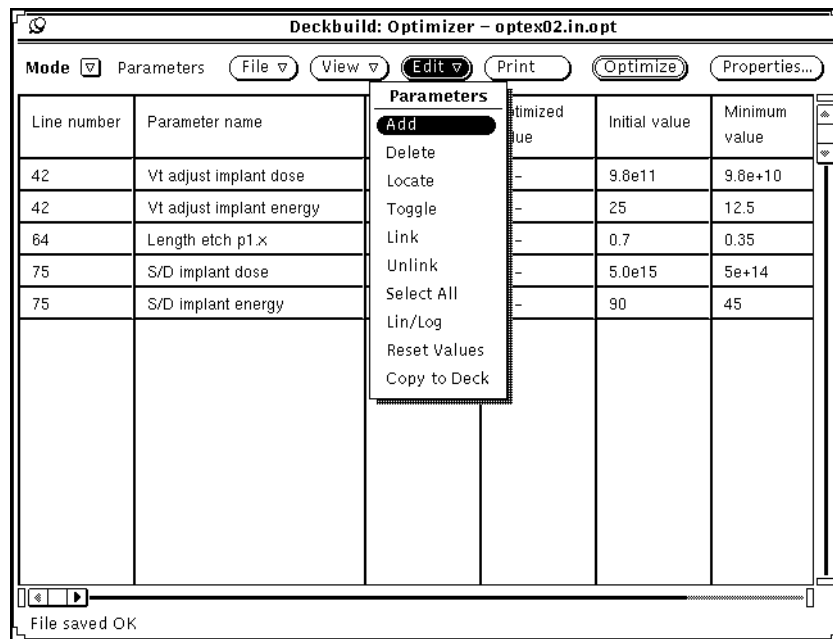


Figure 6-4: Parameter Edit Menu

After selecting **Add**, the **Parameter define** popup appears (Figure 6-5).

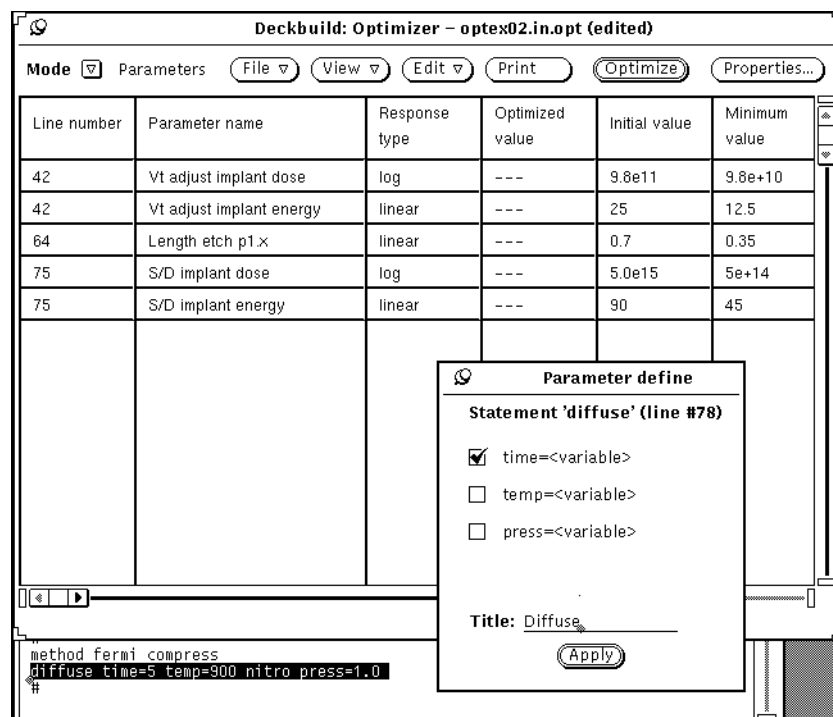


Figure 6-5: Parameter Define Popup

4. Check the checkboxes next to the desired parameter(s) by clicking SELECT next to their names. You can also type in a title. The parameter name on the worksheet are formed by appending the individual parameter names (shown next to the checkboxes) with the title. When you select all the desired parameters, click on **Apply**. A new row is inserted into the **Parameter** worksheet for each selected parameter on the popup (Figure 6-6). The rows are always arranged by line number.

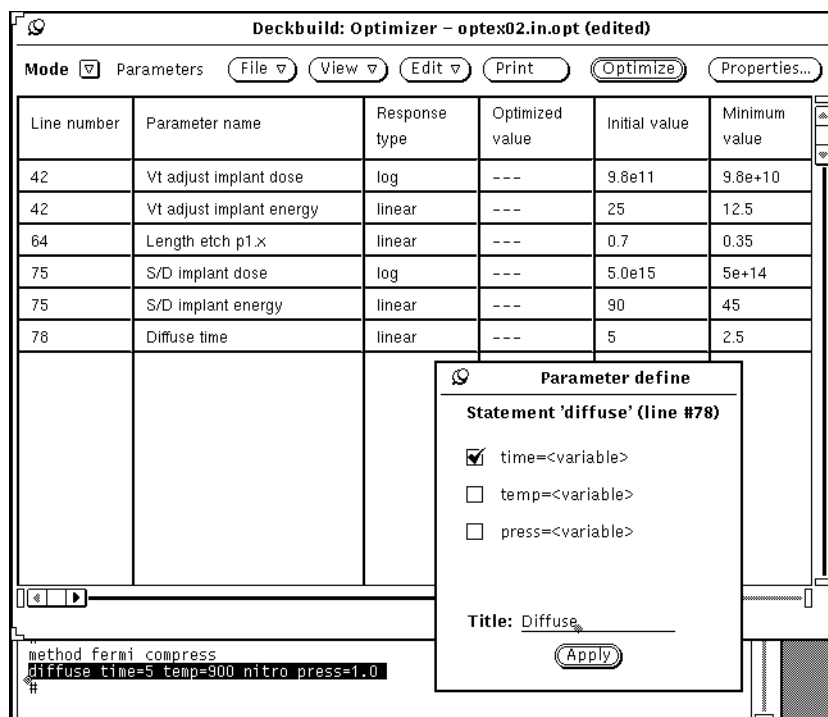


Figure 6-6: Parameters Added to Worksheet

Repeat the process to define parameters on the other lines in the input deck. After adding each parameter, to change its initial value, minimum allowed value, or maximum allowed value. See Section 6.3.3: “Editing A Parameter” for more information.

6.3.2: Deleting Parameters

To delete a parameter:

1. Place the pointer anywhere in the row to be deleted and double-click the **SELECT** mouse button to make a row selection. The row then appears raised (Figure 6-7).
2. Choose **Delete** from the pull-down **Edit** menu to remove the selected row from the worksheet.

The **Delete** operation works on all currently selected rows. More than one row at a time can be deleted. See Section 6.8: “Worksheet Editing” for instruction on selecting more than one row.

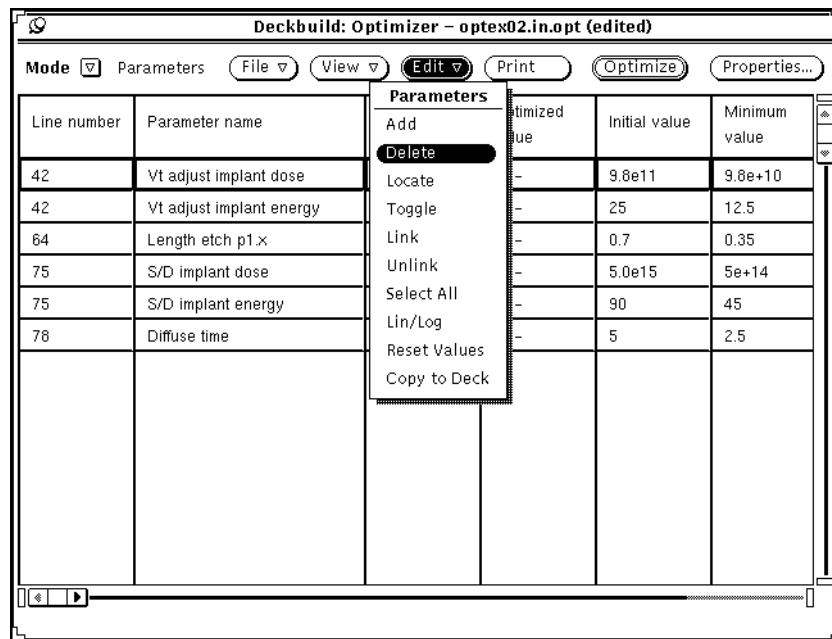


Figure 6-7: Deleting a Parameter

6.3.3: Editing A Parameter

The **Parameter** worksheet allows direct numeric cell value editing, such as **Initial value**, **Minimum value**, and **Maximum value**. See “Parameter Fields” on page 6-11 for an explanation of the fields.

Editing Numeric Values

To edit a numeric cell value:

1. Position the pointer over the cell. The cell will appear indented.
2. Click SELECT once and a text caret appears in the cell.
3. Edit the cell value. The worksheet understands normal keyboard input plus Control-U, which erases the cell contents. Enter the new value, replacing the existing value.
4. Finish the edit by pressing the **Return** key. The new value remains on the worksheet cell.

When the cell is read-only and cannot be edited, a warning message is displayed in the lower left corner of the worksheet. See Section 6.8: “Worksheet Editing” for more information on using the worksheet.

Editing The Response Type

Response type is either **linear** or **log**. Response type should be set to **log** if the extracted results vary logarithmically with the input parameter.

To change the response type:

1. Position the pointer anywhere in the row and double-click the SELECT mouse button to capture the row. The row appears raised.
2. Choose **Lin/Log** from the **Edit** pull-down menu. The selected row’s response type are toggled.

Since **Lin/Log** operates on all selected rows, you can select and toggle multiple rows at the same time.

Editing The Parameter Name

The OPTIMIZER forms the parameter names when entered by appending the parameter name (shown next to the checkboxes on the **Parameter define** popup) with the title field on the popup. By default, the title is the command word (first word) on the line. For example, the parameter dose in an implant statement would be called implant dose on the worksheet. If you type LDD Implant as the title, the parameter name will be called **LDD Implant dose** on the worksheet.

To help distinguish the parameter names, you may need to change them once entered on the worksheet. To change a parameter name:

1. Choose **View→Control...** (Figure 6-8) and the **Parameter control** popup will appear (Figure 6-9).

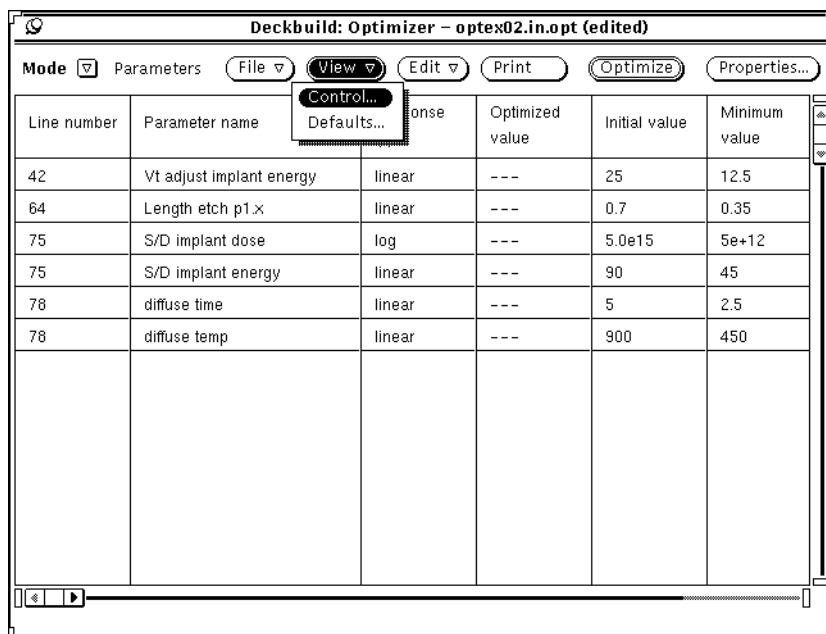


Figure 6-8: Parameter View Menu

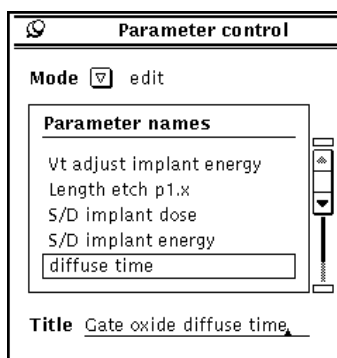


Figure 6-9: Parameter Control Popup, Edit Mode

2. Change **Mode** to **edit** on this popup. The scrolling list contains all the parameter names.
3. Click SELECT over the parameter name that needs to be changed. The parameter name appears under **Title**. Enter in the new parameter name under **Title** and press the Return key.

Once you press Return, the parameter name will be updated on the worksheet and on the scrolling list.

6.3.4: Linked Parameters

Occasionally, it is desirable to “link” two or more parameters together. For instance, a series of several diffusion steps in a row might all use the same HCl percentage or the same pressure. When they are entered as parameters but not linked, each parameter is varied independently during the optimization run. This is not meaningful if the physical diffusions are done in the same environment. When parameters are linked together, they all vary together as one during the optimization run.

Do the following to link parameters together.

1. Select a row to be linked by double-clicking **SELECT** over that row.
2. Select one or more additional rows by single-clicking **ADJUST** over each additional row.
3. Choose **Link** from the **Edit** pulldown menu. **Link** applies to all currently selected rows.

The **Link** operation works by referring the second, third and other linked rows back to the first selected row (counting from the top down). That first row becomes the master row to which all the others are linked (Figure 6-10).

Line number	Parameter name	Response type	Optimized value	Initial value	Minimum value
78	Pressure 1	linear	---	1.00	0.5
79	Drive time	linear	---	60	30
79	Pressure 2->78:Pressure 1	linear	---	1.00	0.5
80	Pressure 3->78:Pressure 1	linear	---	1.00	0.5

```
#
method fermi compress
diffus time=10 temp=900 t.final=1000 nitro press=1.00
diffus time=60 temp=1000 nitro press=1.00
diffus time=10 temp=1000 t.final=900 nitro press=1.00
#
```

Figure 6-10: Linked Parameters

After all rows have been linked, the linked parameter titles change. The titles are appended with an arrow, the line number, and the master parameter name. For example, Pressure 2 becomes Pressure2->78:Pressure 1.

Linked parameters all share the same response type and optimized, initial, minimum, and maximum values. The initial, minimum, and maximum values are taken from the first, parent parameter, but are not updated on the child parameters. This preserves their independent settings in case the parameters are unlinked later. The optimized value is taken from the master parameter and updated on each of the linked parameters during optimization.

It is also possible to link in a new parameter to a set of already-linked parameters. Select any one of the already linked parameters, plus the new parameter, and choose **Link** again on the menu. The OPTIMIZER knows how to follow the sequence of links.

If the master parameter is deleted, all links to it are automatically removed. If a linked parameter is deleted, it does not affect any of the other links. It is important to remember that links are not saved when the optimizer setup is stored to a file. The links have to be set up again if an optimizer file is reloaded. To unlink a parameter:

1. Select the row to be unlinked by double-clicking **SELECT**.
2. Select additional rows, if desired, by single-clicking **ADJUST**.
3. Choose **Unlink** from the **Edit** menu.

6.3.5: Parameter Defaults

The **Parameter** worksheet has built-in values to determine the appropriate response type and minimum and maximum values for a new parameter.

By default, parameters are of response enter **log**, if their initial value is above $1e+10$ and min/max values are $\pm 50\%$ for linear parameters, and ± 1 decade for log parameters.

You can change response type and min/max values after adding parameters. These defaults only help make adding parameters quicker and easier.

Do the following to change the default response type and min/max ranges used when adding a new parameter.

1. Select **View**→**Defaults...** and the **Parameter defaults** popup will appear (Figure 6-11).

Figure 6-11: Parameter Defaults

2. Enter the desired new default value(s).
3. Click on **Apply** when finished. If you want to save the values as permanent defaults, click on **Save**.

6.3.6: Copying Parameters To The Deck

The **OPTIMIZER** automatically fills in the **Optimized value** column on the **Parameter** worksheet as it runs. If the **OPTIMIZER** converges successfully and retaining the optimized parameter values in the input deck is desired, select **Edit**→**Copy to Deck**.

Copy to Deck copies all optimized parameter values back into the deck in place of the former values. Save the input deck (using the **DeckBuild File** menu) to generate the permanent changes.

6.3.7: Enabling/Disabling Parameters

The **OPTIMIZER** allows the disabling of parameters, which are then ignored during optimizations. Disabling is often useful when a large number of parameters have been entered. But, you may want to try freezing one or more at certain values without deleting them from the worksheet. To disable a parameter:

1. Position the pointer anywhere on the row to be disabled and double-click the **SELECT** mouse button to select that row. The row is selected.
2. Choose **Toggle** from the **Edit** pull-down menu. The selected row is grayed out on the worksheet, and its values are frozen.

To enable a disabled parameter, follow the same procedure starting with a disabled row.

Note: During the optimization run, the OPTIMIZER uses whatever entered optimized value for disabled parameters. If no optimized value exists, the initial value is used.

6.3.8: Folding Columns

The OPTIMIZER allows folding, or hiding, any column or columns on the worksheet. This can be useful when several columns on opposite sides of the worksheet need to be seen simultaneously, without having to scroll horizontally back and forth. To fold a worksheet column:

1. Choose **Control...** from the View pulldown menu, and the **Parameter control** popup shown in Figure 6-12 appears.

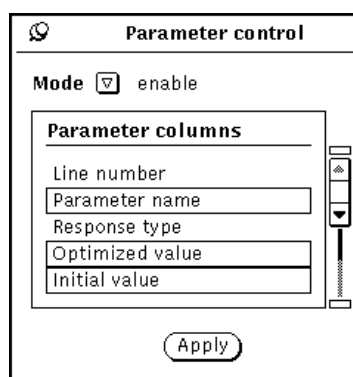


Figure 6-12: Parameter Control Popup, Enable Mode

2. Change **Mode** to **enable** on this popup.
3. Click **SELECT** on any columns in the scrolling list that you want to fold. Columns that remain selected on the scrolling list are shown; the others are folded and are not shown. Click on **Apply** to apply the changes.

Parameter Fields

- **Line number** — The line number in the input deck on which the parameter is used. The line number is updated automatically when the input deck is edited.
- **Parameter name** — The name of the parameter. The name can be redefined from the **Parameter** control popup.
- **Response type** — Either **linear** or **log**. The response type determines how far the OPTIMIZER swings the parameter value. Log type parameters are swung further than linear parameters, and should be used for parameters which have a logarithmic effect on the extracted results.
- **Optimized value** — The value determined by the OPTIMIZER to achieve the best fit to the target(s).
- **Initial value** — The initial value of the parameter, taken from the input deck.
- **Minimum value** — The minimum value that the OPTIMIZER is allowed to use.
- **Maximum value** — The maximum value that the OPTIMIZER is allowed to use.

Note: As an aid to optimization, the Optimizer uses Optimized value as each input parameter's initial value after the Optimizer has been run once. To use the Initial value once again, choose Reset Values on the Parameter Edit menu. The optimized values are deleted on the Parameter worksheet.

6.4: Targets

A target is the value of any valid extract statement in the input deck. **EXTRACT** statements are special commands understood by **DECKBUILD** that extract certain properties of the simulated device. For example, you can extract properties, such as material thickness, net doping versus depth, and junction depth for process simulation and extract I-V curves and parameterized values, such as V_t , Θ , and $DIBL$ for device simulation. **EXTRACT** also allows you to construct your own routine to perform customized manipulation of points and curves.

DECKBUILD provides full popup-driven generation of the extract statements. For more information, see Chapter 5: “DeckBuild:Extract”.

For purposes of the **OPTIMIZER**, **EXTRACT** statements come in two flavors. Those that extract a single value (like V_t) called point targets, and those that extract an entire curve (like V_g vs. I_d) called curved targets.

6.4.1: Adding A Target

To add a target, follow these four steps:

1. Display the **Target** worksheet by setting **Mode** to **Targets**. To do this, position the pointer over **Mode**, press the **MENU** mouse button, and select **Targets**. The **Target** worksheet will then appear (Figure 6-13).

Line number	Target name	Target type	X value	Target value	Optimized value
130	Vt curve	log	0	9.10474e-14	---
130		log	0.5	5.94327e-09	---
130		linear	1	3.6632e-06	---
130		linear	1.5	9.51657e-06	---
130		linear	2	1.42802e-05	---
130		linear	2.5	1.82314e-05	---
130		linear	3	2.16368e-05	---
130		linear	3.5	2.46356e-05	---
130		linear	4	2.72568e-05	---
130		linear	4.5	2.9631e-05	---
130		linear	5	3.17689e-05	---

Figure 6-13: Target Worksheet

2. Select (highlight) the line in the input deck that contains the target you want to add. To do this, position the pointer over the line, triple-click the **SELECT** mouse button to capture the entire line. The **OPTIMIZER** also accepts lines that have only a word or any character(s) selected.
3. Choose **Add** from the **Edit** pull-down menu. To do this, move the pointer over the **Edit** button, click **MENU** and select **Add** (Figure 6-14).

A new row is inserted into the **Target** worksheet for the selected extract statement (Figure 6-15). The rows are always ordered by line number.

4. Enter a target value for the new target. Target value entry depends on the extract value type (point or curve). Both types of target value methods are explained on the following pages.

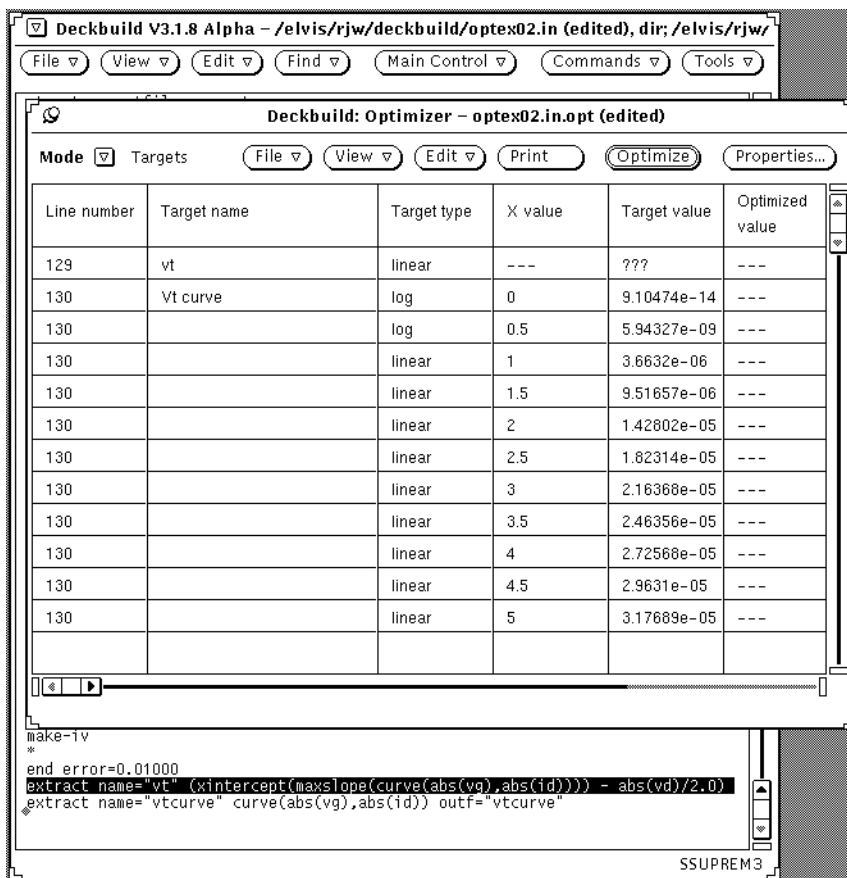


Figure 6-14: Targets Added to Worksheet

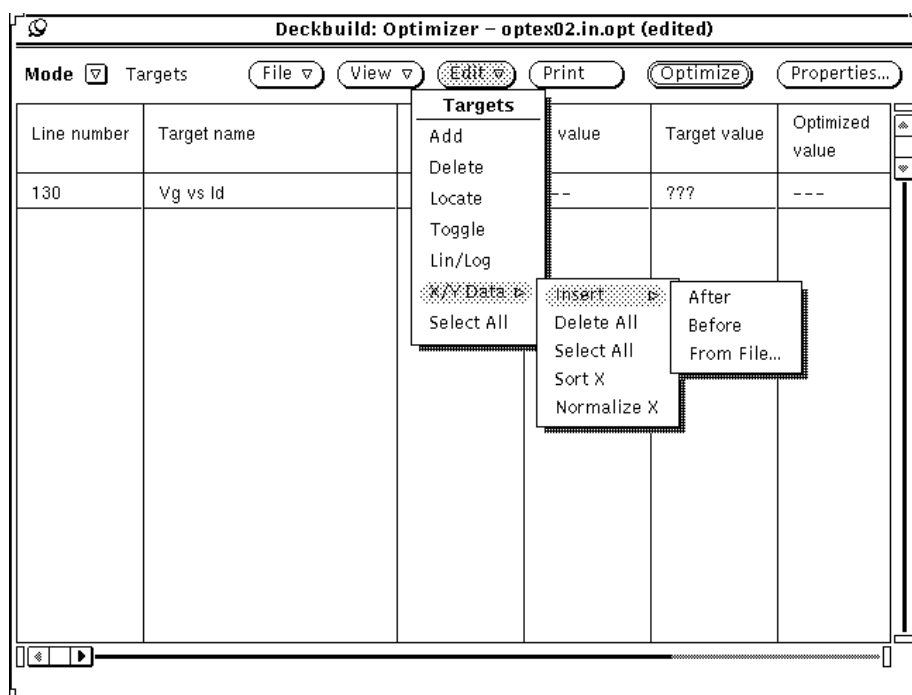


Figure 6-15: Target X/Y Data Menu

Defining a Point Value Target

For point targets, enter the target value directly into the **Target value** column. See Section 6.4.3: “Editing A Target” for more information. Repeat this process to define other targets in the input deck.

Defining a Curve Value Target

For curved targets, you must enter one or more x/y coordinate pairs. The **X value** column on the worksheet corresponds to the x axis of the extracted curve. The **Target value** column corresponds to the y axis. You can do any of the following to enter x/y coordinate pairs for curved targets.

- Insert pairs into the worksheet,
- Read in x/y data from a TONYPLOT data format file
- Do a combination of both.

Creating A Curved Target From Scratch

Do the following to insert an x/y value pair.

1. Position the pointer anywhere in the target row and double-click the SELECT mouse button to select that row. The row then appears raised.
2. Choose **Edit**→**X/Y Data**→**After** (Figure 6-15). A new row will be inserted after the selected row. You can insert additional data rows either before or after any other data row.

Note: Only the first row of a curved target shows the target name (Figure 6-16).

3. Enter **X value** and **Target value** values for the new row.

Repeat this process to define the entire curved target.

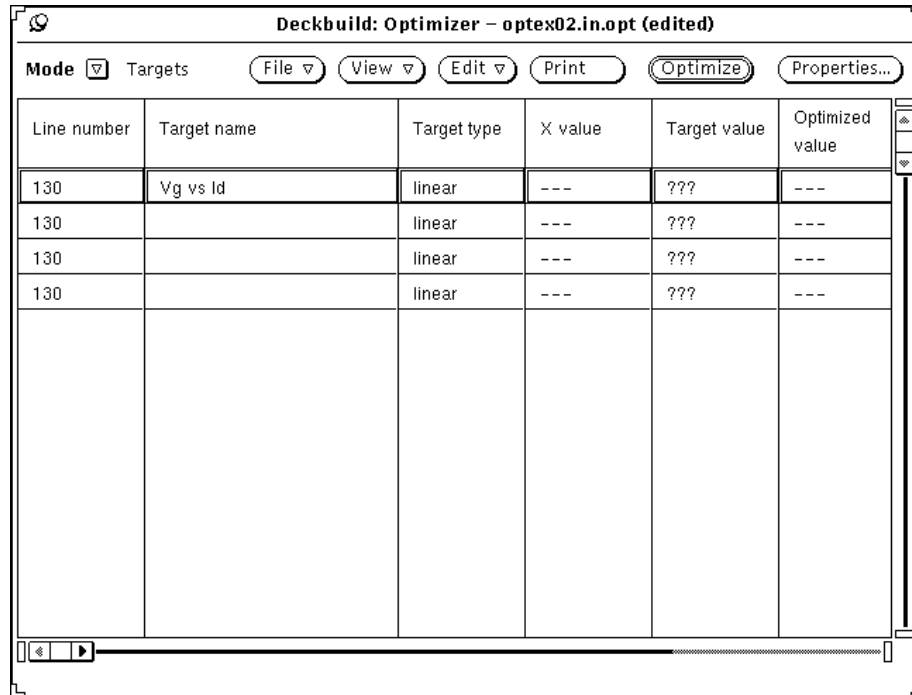


Figure 6-16: A Curved Target

Creating a Curved Target From a Data File

When a TONYPLOT data format file is on the disk that you want to use as an optimization target, the OPTIMIZER allows it to be read directly into the worksheet.

To create a curved target from a data format file, follow these steps:

1. Position the pointer anywhere in the target row, and double-click the **SELECT** mouse button to select that row. The row will then appear raised.
2. Choose **Edit**→**X/Y Data**→**From File...** and the **Optimizer X/Y Target Data** popup will appear (Figure 6-17).
3. Enter in the directory and filter or double-click on directory names in the scrolling list to change directory. To load a file, double-click on the file name in the scrolling list or select the file name in the list and click on **Load**. If the file is in the correct format, existing data rows for the curved target are removed and replaced by rows constructed from the X/Y data in the data file.

To quickly and easily create a TONYPLOT data format file for the target, specify a file name on the `extract` statement. For example:

```
extract curve(depth, abs(net)) outf="net_doping.dat"
```

EXTRACT saves the file in TONYPLOT data format. Run the deck through using DECKBUILD, then load in the saved data file into the worksheet, add/delete rows and edit the values to create the desired target curve.

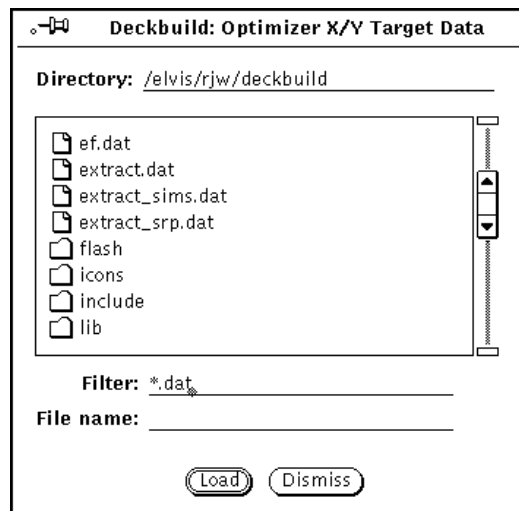


Figure 6-17: X/Y Target Data Popup

6.4.2: Deleting A Target

To delete a target:

1. Position the pointer anywhere in the row to be deleted and double-click the **SELECT** mouse button to select that row. The row will then appear raised (Figure 6-18).
2. Choose **Delete** from the **Edit** pull-down menu. The selected row will be removed from the worksheet.

The **Delete** operation works on all currently selected rows, more than one row at a time can be deleted. See Section 6.8: “Worksheet Editing” for procedures on selecting more than one row.

For curved targets, **Delete** deletes only the selected x/y data point(s). To delete the entire target at once, select any row in the curve to be deleted (as in step 1 above) and choose **Delete All** from the **X/Y Data** menu (Figure 6-19).

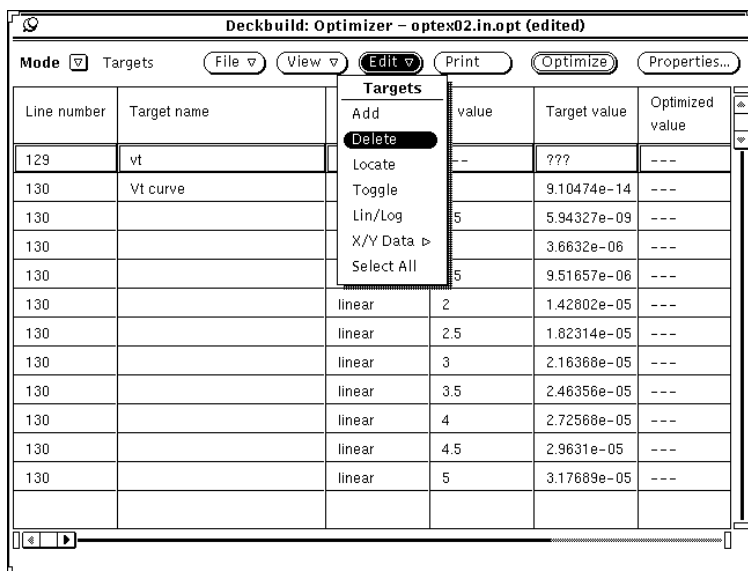


Figure 6-18: Deleting a Target

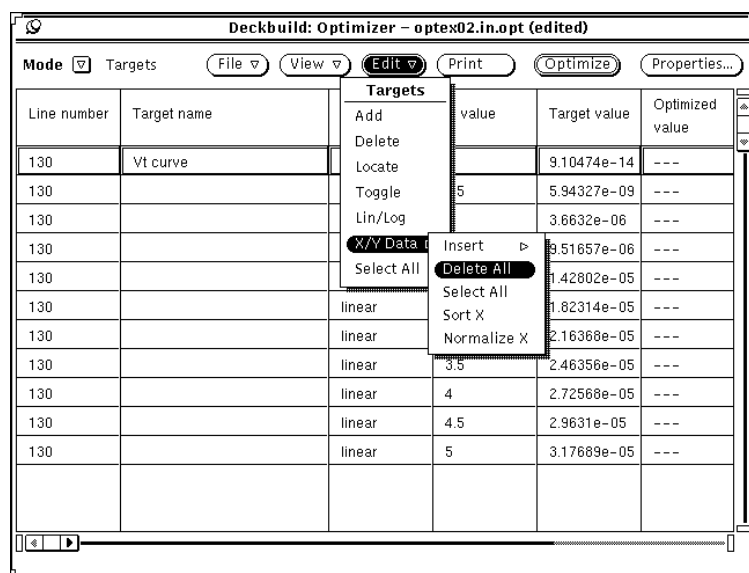


Figure 6-19: Deleting a Curved Target

6.4.3: Editing A Target

The **Target** worksheet allows direct edit of numeric cell values, such as **X value**, **Target value**, and **Weight**. See “Target Fields” on page 6-19 below for an explanation of the fields.

Editing Numeric Values

To edit a numeric cell value:

1. Position the pointer over the cell. The cell then appears indented.
2. Click SELECT once. A text caret appears in the cell.
3. Edit the cell value. The worksheet understands normal keyboard input plus Control-U, which erases the cell contents. Enter the new value.
4. Finish the edit by pressing the Return key. The new value remains in the worksheet cell.

If the cell is read-only and cannot be edited, a warning message is displayed in the lower left-hand corner of the worksheet.

See Section 6.8: “Worksheet Editing” for more information about using the worksheet.

Editing the Target Type

Target type is either **linear** or **log**. Set the response type to **log** if the target value is extremely small to reduce the target’s apparent error. For extremely small targets, a very small absolute difference between the target and the extracted data could result in an extremely large percentage error (1e-12 vs. 1e-11 is a 900% error). Setting target type to **log** tells the OPTIMIZER to take the log of the result and target before computing the percentage error.

To change the target type:

1. Position the pointer anywhere in the row and double-click the SELECT mouse button to capture that row. The row will appear raised.
2. Choose **Lin/Log** from the **Edit** pulldown menu. The selected row’s response type is toggled.

Since **Lin/Log** operates on all selected rows, you can select multiple rows and toggle at the same time.

Editing The Target Name

The OPTIMIZER fills in the target name, by default, when it is first added to the worksheet. The target name is taken from the **name** token in the extract statement. You may want to change the target names to help distinguish them. To change a target name:

1. Choose **Control...** from the **View** pulldown menu and the **Target control** popup will appear (Figure 6-20).

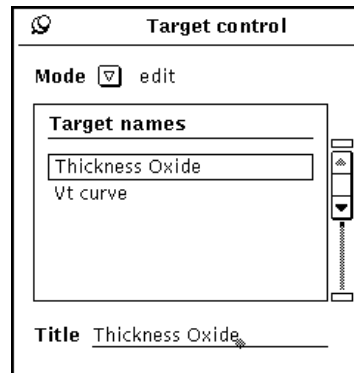


Figure 6-20: Target Control Popup - Edit Mode

2. Change **Mode** to **edit** on this popup. The scrolling list contains all the target names.
3. Click SELECT over the target name to be changed. The target name appears under **Title**. Enter the new target name under **Title** and press Return.

When you press Return, the target name will be updated on the worksheet and on the scrolling list.

6.4.4: Enabling/Disabling Target

The OPTIMIZER allows you to disable targets. Disabled targets are ignored during the optimization. Disabling is often useful when a large number of targets have been entered. There is a need to concentrate on an important subset of targets without deleting the remaining targets from the worksheet.

To disable a target:

1. Position the pointer anywhere in the row that needs to be disabled and double-click the SELECT mouse button on the row. The row is selected.
2. Choose **Toggle** from the **Edit** pulldown menu.

The selected row is grayed out on the worksheet, and its values are be frozen. To enable a disabled target, follow the same procedure precisely beginning with a disabled row. During an optimization run, the OPTIMIZER does not monitor or count the error terms from disabled targets.

6.4.5: Folding Columns

The OPTIMIZER allows you to fold, or hide, any column or columns on the worksheet. This may be useful when there is a need to see several columns on opposite sides of the worksheet without having to horizontally scroll back and forth.

To fold a worksheet column:

1. Choose **Control...** from the **View** pulldown menu and the **Target control** popup will appear (Figure 6-21).

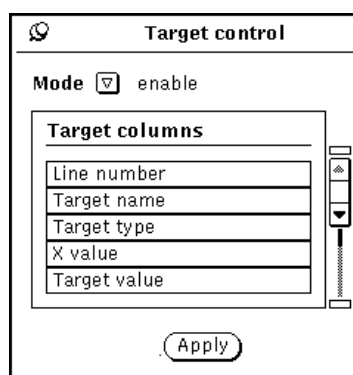


Figure 6-21: Target Control Popup - Enable Mode

2. Change **Mode** to **enable** on this popup.
3. Click **SELECT** on any columns that are to be folded in the scrolling list. Columns remaining selected on the scrolling list are shown. The others are folded and not shown. Press **Apply** to apply changes.

Target Fields

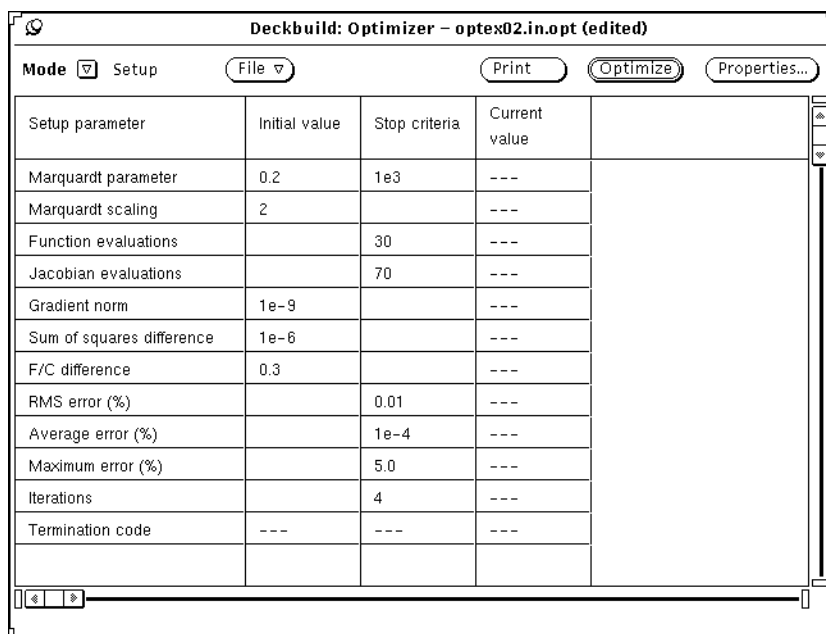
- **Line number** — The line number that the target is on in the input deck. The line number is automatically updated when the input deck is edited.
- **Target name** — The name of the target. You can redefine the name using the **Target control** popup.
- **Target type** — Either **linear** or **log**. The target type determines how the **Optimizer** computes the error term. Log type targets should be used where the target value is extremely small.
- **X value** — The X axis value of an x/y coordinate pair. It is only used by curved targets.
- **Target value** — The optimization target value (Y axis value for curved targets).
- **Optimized value** — The current value of the extracted (simulated) data.
- **Error (%)** — The percentage error between **Target value** and **Optimized value**.
- **Weight** — The target weighting factor. The weight is multiplied by the actual error to determine the apparent error. A value of 1 indicates that the **Optimizer** shall use the actual error, a value of 0.5 indicates that it should use only half the actual error. Values between 0 and 1 make the target less sensitive. Therefore, it is subject to wider error tolerance, while values greater than 1 make it more sensitive and force the simulated data to smaller error tolerance. For instance, a weight of 2.0 with a setup maximum error of 5.0% would force a target to have a maximum actual error of 2.5% to achieve convergence.

Note: The error percentage shown on the Targets worksheet is the actual error percentage before weighting.

6.5: Setup

6.5.1: Overview

The **Setup** worksheet controls a number of constants used by the OPTIMIZER, such as the maximum number of iterations and evaluations and several different convergence criteria. The default values are almost always adequate. The **Setup** worksheet is displayed by setting **Mode** to **Setup** (Figure 6-22).



Setup parameter	Initial value	Stop criteria	Current value	
Marquardt parameter	0.2	1e3	---	
Marquardt scaling	2		---	
Function evaluations		30	---	
Jacobian evaluations		70	---	
Gradient norm	1e-9		---	
Sum of squares difference	1e-6		---	
F/C difference	0.3		---	
RMS error (%)		0.01	---	
Average error (%)		1e-4	---	
Maximum error (%)		5.0	---	
Iterations		4	---	
Termination code	---	---	---	

Figure 6-22: Setup Worksheet

The most important setup criteria are as follows:

- **Maximum error** — Determines the maximum allowable error for any target to achieve convergence.
- **Iterations** — Determines how many total Marquardt iterations the optimizer is allowed to perform. This value counts only the Marquardt iterations, each consists of one or several sub-iterations. The actual number of simulation calculations is always greater than this value.

The default value of **Maximum error** and **Iterations** is 5.0% and 4 respectively.

6.5.2: Editing Setup Values

To edit a setup cell value:

1. Position the pointer over the cell. The cell then appears indented.
2. Click SELECT once. A text caret appears in the cell.
3. Edit the cell value. The worksheet understands normal keyboard input plus Control-U, which erases the cell contents. Enter the new value.
4. Finish the edit by pressing Return. The new value remains in the worksheet cell.

When the cell is in read-only, a warning message is displayed in the lower left-hand corner of the worksheet. See Section 6.8: “Worksheet Editing” for more information on using the worksheet.

6.5.3: Saving The Setup

To modify the setup data and to save for future runs, choose **Save Setup** from the **File** menu. The setup information is saved in \$HOME/.masterrc.

6.6: Graphics

6.6.1: Overview

The OPTIMIZER maintains a run-time graphics visualization system that allows for tracking the progress of the OPTIMIZER over its iterations, and to evaluate its current status. To access graphics, set the **Mode** to **Graphics** (see Figure 6-23).

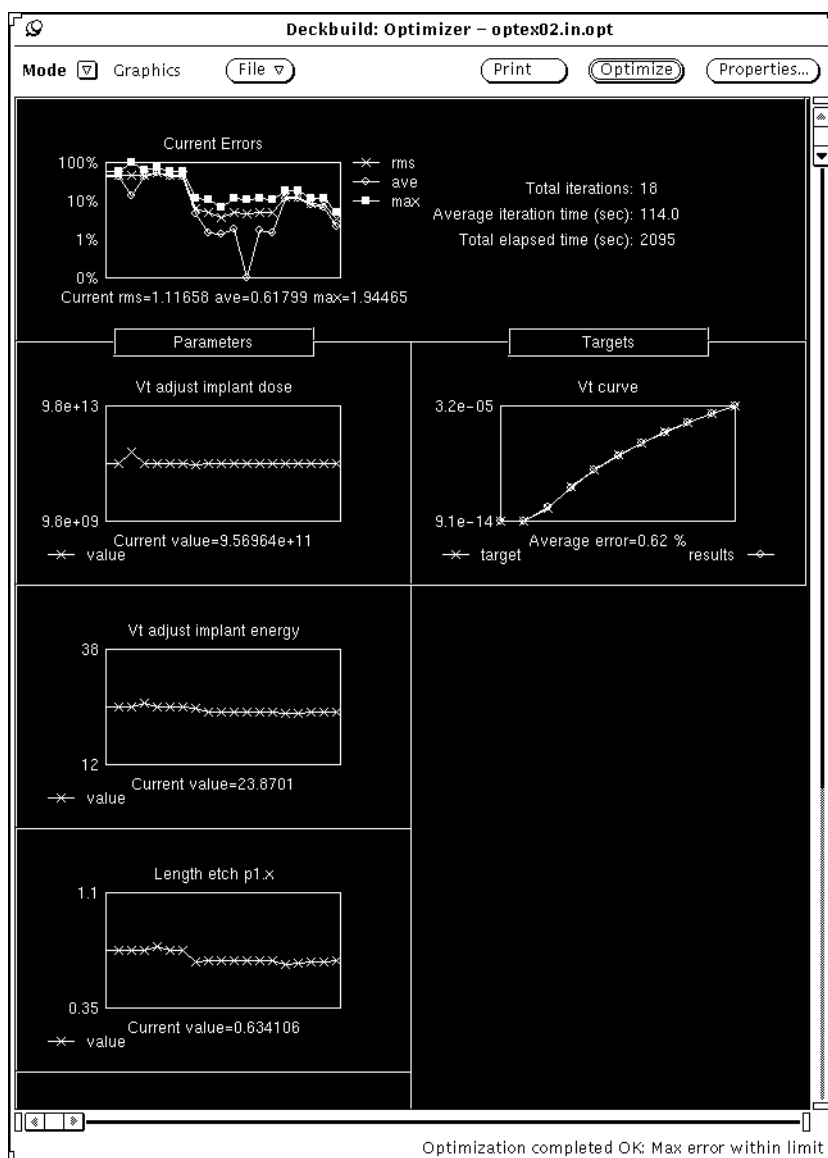


Figure 6-23: A Curved Target

The **Current Errors** section at the top of the window maintains the values of RMS error, average error, and maximum error over all targets. It also displays the total number of iterations so far, the average iteration time, and the total elapsed time. The total elapsed time increments in real time as the optimization runs. The number of iterations counts the number of simulation computations, not Marquardt iterations defined on the **Setup** worksheet.

The **Parameters** column keeps track of the input parameter values. Each rectangle represents, vertically, the allowable range of the parameter (its minimum to maximum value), while the horizontal axis represents the number of iterations. The parameter values are displayed for all iterations to date.

The **Targets** column shows the target values, the current simulated values, and an error term. For single-point targets, the target is drawn as a horizontal line, and simulated values plus error terms are drawn for all iterations to date. For curved targets, the target curve and the simulated results curve are plotted plus the current average error over the entire curve.

This **Graphic** iteration history information is maintained until another optimization run is started or until any target or parameter is added or deleted.

6.7: Results

6.7.1: Overview

An optimized results list is maintained by the OPTIMIZER to show the input parameter values and target results for each iteration. To display the **Results** worksheet, set the **Optimizer's Mode** to **Results**. Figure 6-24 shows a typical **Results** worksheet.

Iteration	Vt adjust implant dose	Vt adjust implant energy	Length etch p1.x	S/D implant dose	S/D implant energy	Vt curve
1	9.8e+11	25	0.7	5e+15	90	err=32%
2	2.34651e+12	25	0.7	5e+15	90	err=7.3%
3	9.8e+11	25.7906	0.7	5e+15	90	err=34%
4	9.8e+11	25	0.722136	5e+15	90	err=42%
5	9.8e+11	25	0.7	1.56825e+16	90	err=32%
6	9.8e+11	25	0.7	5e+15	92.846	err=32%
7	9.22232e+11	24.6085	0.632424	2.36997e+13	86.322	err=1.7%
8	9.32321e+11	23.9048	0.636331	5e+12	83.6261	err=0.36%
9	9.46142e+11	23.9048	0.636331	5e+12	83.6261	err=0.31%
10	9.32321e+11	23.9839	0.636331	5e+12	83.6261	err=0.47%
11	9.32321e+11	23.9048	0.638544	5e+12	83.6261	err=0.0094%
12	9.32321e+11	23.9048	0.636331	5.6055e+12	83.6261	err=0.45%
13	9.32321e+11	23.9048	0.636331	5e+12	83.9107	err=0.35%

Optimization completed OK: Max error within limit

Figure 6-24: Results Worksheet

The left column displays the iteration number, starting from 1. The other columns display the input parameters from the **Parameters** worksheet followed by targets from the **Targets** worksheet.

Input parameter columns show the input parameter values used for each iteration. Target columns show the extract values obtained for the parameter values, or if a curved target, the average error for the entire curve.

The **Results** worksheet is automatically updated at run-time and a new row is added every time an iteration finishes. If any of the parameter or target names are changed, the changes are reflected in the **Results** worksheet.

6.7.2: Sensitivity

The **Results** screen also displays a sensitivity calculation performed as the optimization progresses.

The sensitivity of each target is displayed next to the target's value column. The sensitivity is calculated as the percentage change in the target value divided by the percentage change in a single input value.

The sensitivity displayed on a given row is the result of changing the single input parameter on that row, which differs from the baseline calculation (the topmost result row).

Since the OPTIMIZER first calculates the baseline values and performs the sensitivity calculation on each parameter in turn, thereafter trying to vary all parameters as necessary, the sensitivity results are displayed only on the second through *n*th row of the results worksheet, where *n* is the number of parameters plus one.

6.8: Worksheet Editing

6.8.1: Overview

The **Worksheet** allows for editing cells and typing values directly into the cell contents. Some cells, however, (depending on the worksheet mode) are read-only and cannot be edited.

6.8.2: Numeric Values

To edit a numeric cell value on a worksheet:

1. Position the pointer over the cell. The cell appears indented.
2. Click **SELECT** once. A text caret appears in the cell (Figure 6-25).
3. Edit the cell value. The worksheet understands normal keyboard input, plus Control-U, which erases the cell contents. Enter the new value.
4. Finish the edit by pressing Return. The new value remains in the worksheet cell.

When the cell is read-only, a message footer is displayed in the lower left corner of the worksheet.

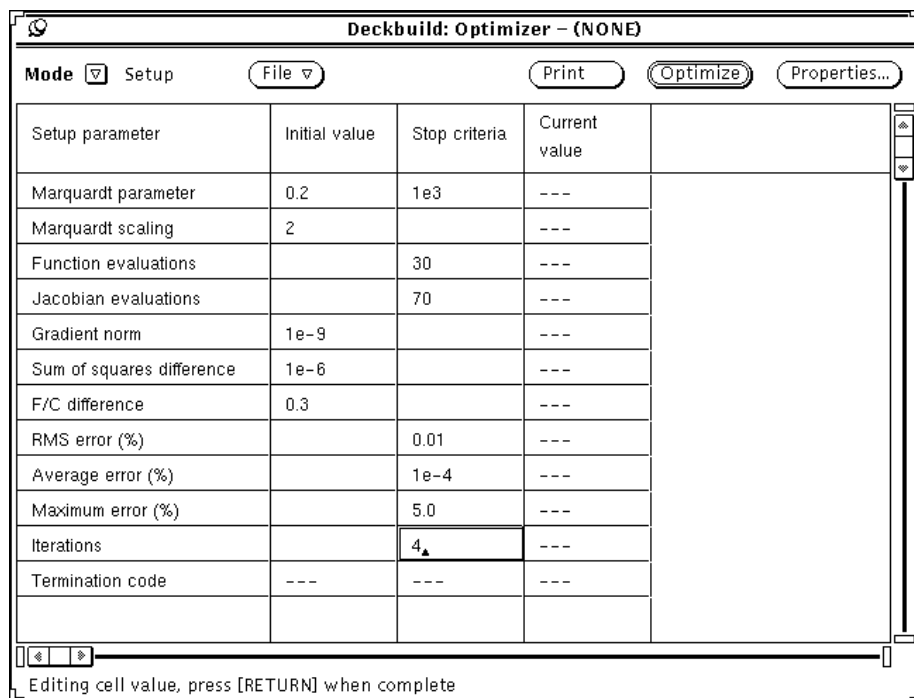


Figure 6-25: Editing a Worksheet Cell

6.8.3: Selecting Rows

To select a worksheet row, position the pointer anywhere over the row and double-click the **SELECT** mouse button. To select a second or any further row (extending the range of the selection), click **ADJUST** over each additional row to be selected. To deselect a row, position the pointer over the selected row and click **ADJUST**. **ADJUST** toggles the selected state. To deselect all rows, press **SELECT** once anywhere in the worksheet.

Note: Selections are lost when the worksheet mode is changed.

6.8.4: Mouseless Operation

The worksheet supports mouseless operation. You can traverse rows and columns of the worksheet with the arrow keys, plus the Home, End, PgUp, and PgDn keys. You can edit a worksheet by pressing Return in a cell.

6.9: File I/O

6.9.1: Overview

When an optimization run is finished, the OPTIMIZER allows storage of all pertinent data to disk. You can load an optimization file at any later point to initialize the OPTIMIZER with all the saved target and parameter data. The Setup data is stored independently. For procedures on Setup data, see Section 6.5: “Setup”.

6.9.2: Creating A New File

To create a new optimization file:

1. Move the pointer to the **File** menu button, press the MENU mouse button, and select **Store as New File...** (Figure 6-26). The **Store** popup will then appear (Figure 6-27).

Note: The current directory is listed on the Store popup.

2. Move the pointer into the **Store** window and enter the name of the preferred file. By default, the file name is the name of the current input deck with .opt appended.
3. Click on **File→Store as New File** and an ASCII format optimization file will be saved. This file contains all parameter and target information.

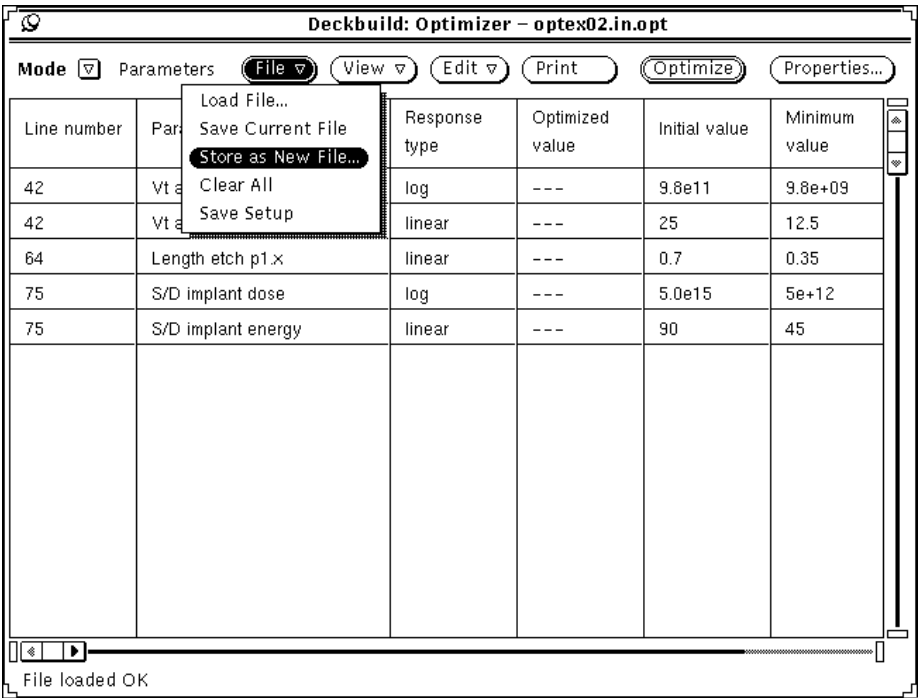


Figure 6-26: Storing a New File

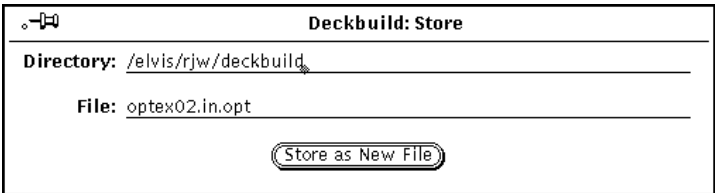


Figure 6-27: Store Popup

Note: Parameter links are not saved in the optimization file. Any linked parameters have to be relinked when the optimization file is loaded.

6.9.3: Working With An Existing File

Any previously saved optimization file can be loaded and modified or stored under another name to create a new file, leaving the original intact.

To load a file into the OPTIMIZER:

1. Choose **File→Load File...** on the **Optimizer** window and the Optimizer Load Popup shown in Figure 6-28 will appear.

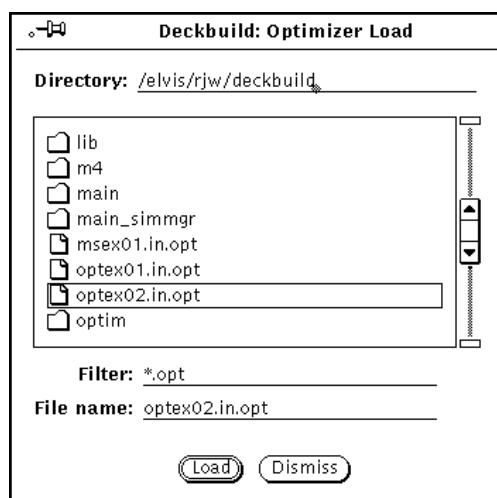


Figure 6-28: Optimizer Load Popup

2. Adjust **Filter**, if necessary, to filter out all but the file names of interest. By default, the filter is *.opt. Press Return to apply the filter.
3. Change the directory, if necessary, by double-clicking on a directory in the scrolling list or by entering a directory name under **Directory**.
4. Once you have found the file you want to load, load it by double-clicking on it in the scrolling list. You can also load the file by clicking on **Load**.

Note: The file to be loaded must correspond to the current input deck. If the information in the optimizer file does not match the contents of the input deck, the OPTIMIZER will refuse to load the file and an error message appears. For this reason, always save the input deck and its optimizer file at the same time.

6.9.4: Saving The Existing File

You can now edit the OPTIMIZER data and save those changes back to the existing OPTIMIZER file or save them as a new file.

To save the changes to the existing file, choose **File→Save Current File** on the OPTIMIZER window.

Be sure to save the input deck as well, if it has been edited. Save the input deck using DECKBUILD's **File** menu.

6.10: Printing

6.10.1: Overview

To print any of the worksheets or graphics data:

1. Press **Properties...** on the **Optimizer** window and the **Optimizer Properties** popup will appear. Change the **Category** in the upper left corner to **Graphics printer** (Figure 6-29).
2. Select the **Destination**, **Type**, **Layout**, and **Page** size of the output. If the destination is **Printer**, also choose the printer queue.

By default, the output will go a file called `print.out` in PostScript format. Indexed, enabled by default, means a unique file name `print.out` is made each time a print is done, so previous files are never overwritten. Click on **Save** to permanently save the settings.

3. Press **Print** back on the **Optimizer** window. The output goes to the file or printer specified in Step 2 and contains data corresponding to the current **Optimizer** mode.

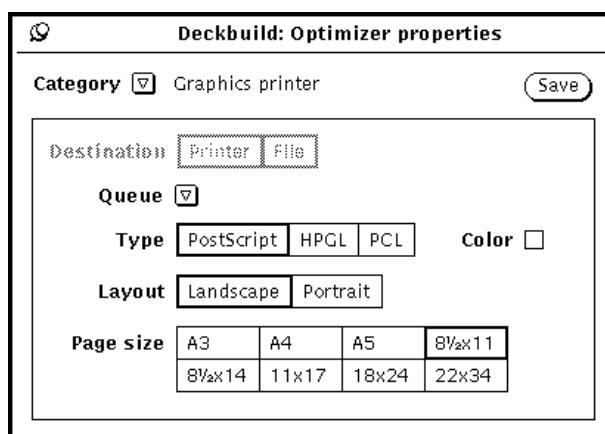


Figure 6-29: Optimizer Properties Popup - Graphic Printer Category

Figures 6-30 and 6-31 shows examples of printed output.

	1	2	3	4
Line number	42	42	64	75
Parameter name	Vt adjust implant dose	Vt adjust implant ener,	Length etch pl.x	S/D implant dose
Response type	log	linear	linear	log
Optimized value	9.56964e+11	23.8701	0.634106	5.51201e+12
Initial value	9.8e11	25	0.7	5.0e15
Minimum value	9.8e+09	12.5	0.35	5e+12
Maximum value	9.8e+13	37.5	1.05	5e+17

	5
Line number	75
Parameter name	S/D implant energy
Response type	linear
Optimized value	83.3429
Initial value	90
Minimum value	45
Maximum value	160

Figure 6-30: Parameters Worksheet Printout

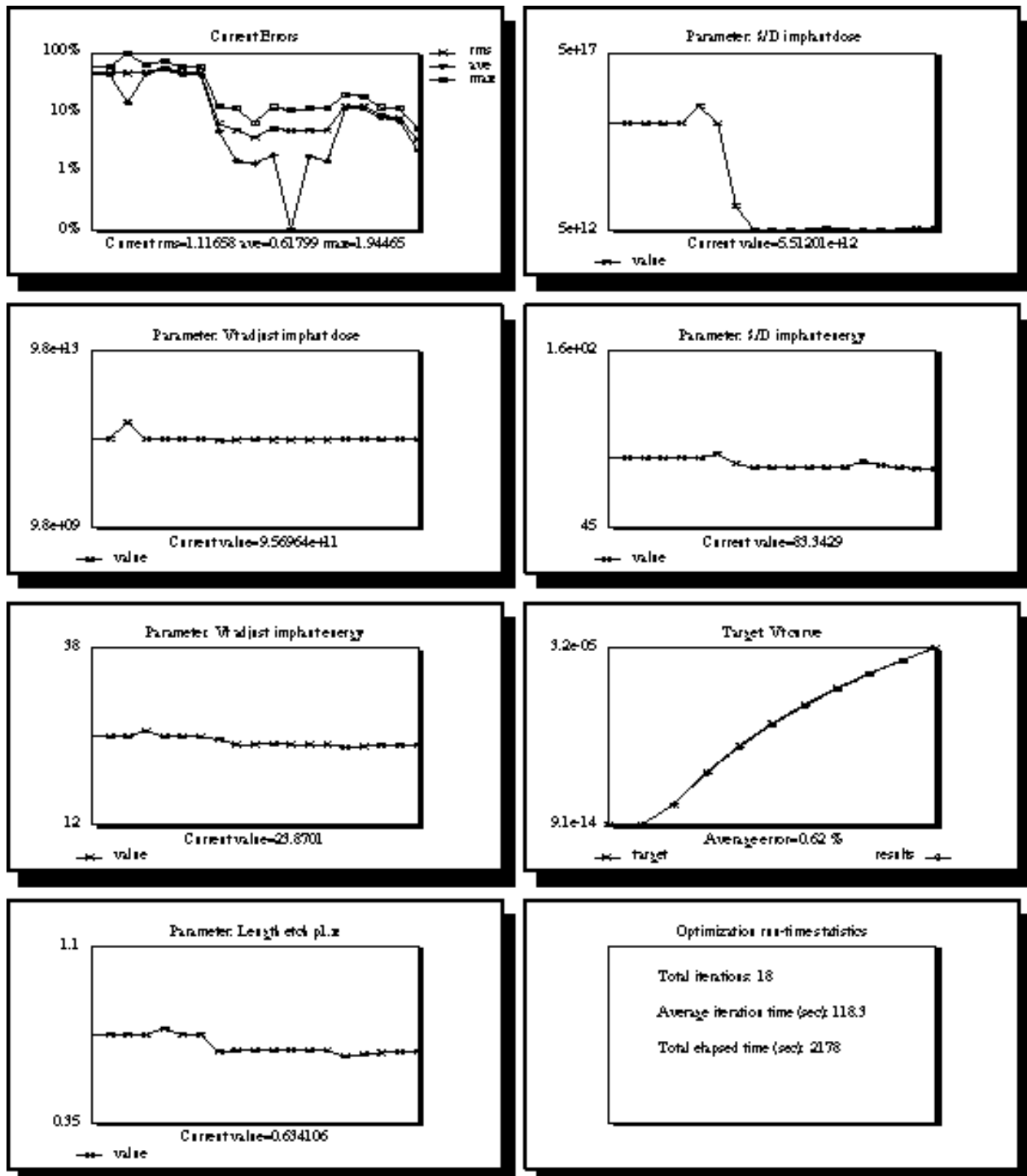


Figure 6-31: Graphic Printout

6.11: Optimization Tuning

This section contains information useful when an optimization run does not converge for some reason. It indicates some common errors and shows how different setup, parameter, and target values are designed to work.

Possible error conditions include the following:

- **Input parameters have little effect on extracted results:** This is usually indicated by very little target movement (as seen on the **Graphics** display) during the optimization. Use different parameters, increase the parameter min/max range, or change the parameter type from **linear** to **log**.
- **Targets conflict with each other:** This is an extremely common problem. For instance, asking the OPTIMIZER to optimize to a thick gate oxide thickness and to a low threshold voltage (that isn't feasible with the thick oxide) is doomed to fail. The best way to get around these types of problems is by carefully choosing targets and target values, plus changing the weighting of some targets to make them more (or less) sensitive than others.

You can make a target less sensitive by setting its weight to a number between 0 and 1. Numbers greater than 1 make it more sensitive (i.e., it must be met with a smaller percentage error).

- **Input parameters hit their minimum or maximum ranges:** This can happen if the initial value is not very close to the value required to achieve optimization. Increase the min/max range of these parameters
- **Failure to converge:** If all the other error conditions are taken care of and the optimization still does not converge, it's often because just a few target points can not be met, or the OPTIMIZER's initial values were too far off the mark.

First, check the error percentages on each target. If most were close but just a few have large error terms, then you might consider decreasing the weight of those targets or changing them from **linear** to **log** type. If those points are part of a curved target, you might consider deleting some or all of them.

On the other hand if the OPTIMIZER fails, try better initial values, decrease the parameter min/max ranges or disable the parameters that have the strongest effect on the results.

Another common reason for convergence failure is a bad curved target. Keep curved targets smooth, without excessive points, and if necessary cut down the min/max range of especially dominant input parameters.

You can also try increasing the number of iterations, but the default value of 4 should almost always be adequate.

Note: As an aid to optimization, the OPTIMIZER uses Optimized value as each input parameter's initial value after the OPTIMIZER has been run once. To use the Initial value once again, choose **Reset Values** on the **Parameter Edit** menu. The optimized values are deleted on the Parameter worksheet.

7.1: Overview

TONYPLOT (version 2) is a graphical post processing tool for use with all Silvaco simulators, and is an integral part of the VWF INTERACTIVE TOOLS. It can operate stand-alone, or along with other VWF INTERACTIVE TOOLS such as DECKBUILD, VWF, or SPDB.

Users of earlier versions of TONYPLOT will find some of the functionality familiar, but other capabilities and the user interface are completely new. All the features that were available in TONYPLOT are still supported in this version. The interfaces are easier to use, however, and the resulting display is also improved. Version 2 of TONYPLOT provides:

- Multiple file loading
- Command language control
- Plot comparison and overlay
- Movie function (replaces the Master tool “Movie”)
- HP4145 emulator
- Process animation
- Cross section profile integration
- Poisson Solver
- Many more user defined properties
- Improved user interface and ease of use
- Improved outline definition and creation
- Faster drawing
- New Structure file features
- User definable setup parameters and set files

7.1.1: Examining Data

TONYPLOT may be used to examine several data files all at once, each in its own plot window. These plot windows can be combined, effectively “overlying” the data sets so that direct comparisons can be made. Plots can be interactively added, deleted and duplicated, overlaid and separated.

Not only does TONYPLOT allow you to display any data file produced by Silvaco tools, but it also provides extensive “tools” for examining these plots and the associated data. For example, it is possible to take cut-line slices through 2D structures, or to integrate a curve to calculate area, or even perform simple electrical simulations on 1D devices.

TONYPLOT allows plots to be rescaled, zoomed and panned. Grids can be added, axes customized, and arbitrary labels drawn on the data. All titles, marks, labels, ranges and so on are automatically set to useful defaults but can all be explicitly set whenever necessary.

The appearance of all plots in TONYPLOT can be totally customized, and there are many “properties” that can be tailored to suit either a certain user, or the requirements of a particular set of data. TONYPLOT supports a wide variety of different printers, and can be setup to print to any size of paper.

TONYPLOT can be used to study the output from Silvaco’s process and device simulators ATHENA and ATLAS. It also plots data from the process database SPDB. When used inside the VWF framework, TONYPLOT can display regression models, response surfaces, scatter plots, histograms SPC charts, pie charts, and more.

7.1.2: Online Help

TONYPLOT includes an online user's manual which can be accessed with a single mouse click. If the manual does not answer a particular problem, help from Silvaco support staff is always available. There is a direct interface to "email" in TONYPLOT, for sending your comments to us.

7.1.3: Terminology

The following words and terms are used throughout TONYPLOT and have certain meanings and relationships. A review of the following summaries will aid in the understanding of subsequent material.

- **Structure:** A structure is that set of data contained within a structure file (doping, geometry, bias points, etc). One structure usually requires one plot, but sometimes two. The same structure can be repeated in many plots.
- **Plot:** A plot is one drawing. A plot can be of one or more structures, but can never exist without a structure. Several plots can show the same structure, which allows the data to be simultaneously examined from two different angles. TONYPLOT is capable of displaying three kinds of plots, each with its own distinct properties: 2d-Mesh plots, Graph plots, and Cross-Section plots.
- **Selected Plot:** Not all plots need be operated on at once. A subset of plots is defined by selecting required plots. Selected plots are indicated by bolded borders, unselected plots have dull borders.
- **View:** — The view is the collection of plots in the main window. It is a global term, and refers to all plots, selected or not.
- **Subwindow:** A subwindow is the area onto which a plot is rendered. Each plot has a unique subwindow, and each subwindow can show only one plot. You can modify the arrangement of subwindows within the view.
- **Display:** Each plot has a display setting, which is the set of parameters that defines how the structures represented in the plot are to be drawn.

7.1.4: Standard Controls

Throughout the use of TONYPLOT, standard controls (Figure 7-1) are often referenced and, in each case, perform the same function. This consistency allows for more efficient use of the popups as familiarity with the standard controls is gained.



Figure 7-1: Standard Controls

The following standard controls are available:

- **Apply:** Almost all popups have an **Apply** button. This button stores the current settings on the popup and usually causes a plot to be updated. Click on this button when satisfied with the items on the popup.
- **Reset:** Many popups also have a **Reset** button. This button restores the settings on a popup to the state they were in when they first appeared, or when the **Apply** button was last clicked on, whichever was most recent. In other words, it sets the popup back to represent the information currently held inside the program. Use this button after modifying popup items, and then deciding to return them to their previous state. However, if **Apply** has been clicked on, **Reset** will not work.
- **Dismiss:** To remove the popup from the screen, click on **Dismiss**. None of the changes made to the popup will be applied. Use this button when the popup is no longer needed and no further changes are required.

- **Save as Defaults:** This button appears as an icon showing data flow from a popup to a disk. Clicking on **Save as defaults** causes the current settings on the popup to first be applied (as if **Apply** were clicked on) and then be saved to a defaults file. For more information on defaults refer to the Defaults section.
- **Load Defaults:** This button appears as an icon showing data flow from a disk to a popup. Clicking on Load defaults causes the current settings on the popup to be loaded from the defaults file. For more information on defaults, refer to the Defaults section.
- **Cycle:** When a choice item has many options, a cycle button appears next to it. Clicking on the **Cycle** button cycles through all the available options one by one, and restarts when the last one is reached. The choice item can be used as normal whether **Cycle** is present, or not. By holding down the Shift key when clicking on a **Cycle** button, it is possible to move backwards through the list, wrapping around to the end when the start has been reached.

7.2: Invoking

7.2.1: Overview

TONYPLOT can be started independently from UNIX, or from other simulator tools such as DECKBUILD, SPDB, or MANAGER. In the case of the simulator tools, starting TONYPLOT is accomplished by selecting the respective command button. The TONYPLOT base window (Figure 7-2) appears if files are not immediately loaded when started.

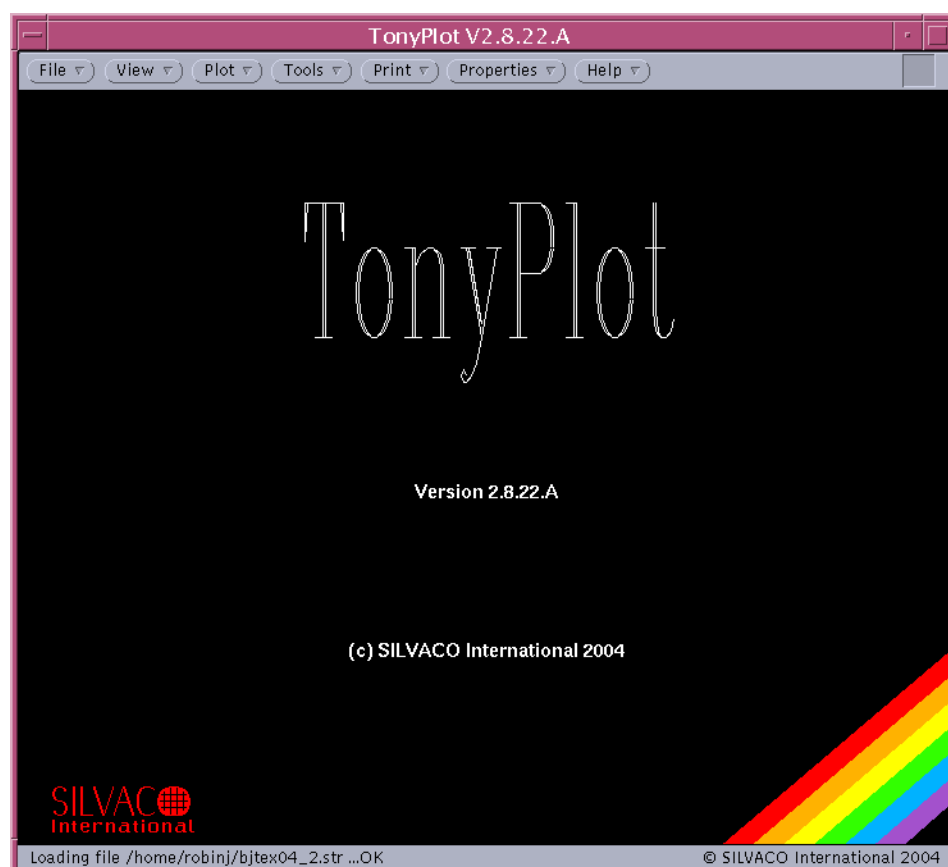


Figure 7-2: TonyPlot Base Window

7.2.2: Starting

To start TONYPLOT at the UNIX prompt enter:

```
% tonyplot
```

This starts TONYPLOT with no data file loaded, and with all options set to their default values.

To load files at the same time TONYPLOT starts, just specify the filenames on the command line. TONYPLOT figures out the type of file and display the data accordingly.

To change some of the options, there are flags available that can also be given on the command line. All of these flags begin with a dash (i.e., -mono), so do not try to load files that have names starting with a dash.

Some of the option flags are position sensitive so that in some cases, the option must precede the file name (when present), and in other cases must follow the file name. However, most of the options are position independent. Some of the options are listed below with a brief description:

-nosort Disables sorting. By default, TONYPLOT sorts all 2D mesh files so that the triangles are ordered. This allows for faster drawing and 3D elevation plots to be drawn.

-print <file> Tells TONYPLOT that you wish to print the data file rather than viewing it on screen. If this is the last argument on the command line, or is preceded by another option flag, a default print file name is used (usually `print.out.0`). Otherwise the proceeding argument is taken as the name of a print file to be created.

-printer <pname> When used with the **-print** option, this tells TONYPLOT to use <pname> as the printer to use. See Printing to determine the printer names allowed and what this means.

-form <fname> When used with the **-print** option, this tells TONYPLOT to use <fname> as the form to use. See Printing to determine the form names allowed and what this means.

-power10 Tells TONYPLOT to show all values that are shown on a log scale as a power of 10. By default, TONYPLOT shows the power index. This can also be controlled with a property (see Properties for more details).

-help Prints out a list of all command line options that TONYPLOT recognizes, including all of the standard **X11**, **XView** and possible **Motif** options as well.

-production Starts TONYPLOT in production mode.

-set <file> Instructs TONYPLOT to load the set filename and restore the display to the condition that TONYPLOT was in when that set file was created. The set file is applied to all files loaded at that point, i.e., all files that preceded this option on the command lines. Files given after this option on the command line are not affected.

-mtitle <maintitle> Overrides the default plot main title and set it to `maintitle` instead. Use single quotes around <maintitle> if you wish to use spaces. All files loaded so far (all that precede this option on the command line) are affected.

-ttitle The same as **-mtitle**, but sets the subtitle instead of the main title.

-overlay Instructs TONYPLOT that all proceeding files are to be loaded and “overlayed” on to the last file loaded. The default is to “add” the new file, i.e., create a new plot. The line `TONYPLOT first.str -overlay second.str` loads in the file “`first.str`” and then overlay “`second.str`” on top of it. This only applies to data files loaded from the command line; files loaded from the **File Loader** are controlled separately.

-add The default load method, and tells TONYPLOT to stop overlaying files as they are loaded in. This only applies to data files loaded from the command line (files loaded from the **File Loader**) are controlled separately.

-ccd Allows the overlay of results of the `deckbuild extract max.conc.file` output on 2D mesh plots. This is used in CCD analysis to determine the potential maximum/minimum

The **-da**, **-st** and **-bin** options to TONYPLOT are all optional. When TONYPLOT attempts to load a file, it automatically works out the format of the data and load it in the correct manner. The options override this action however, if this is needed.

Here are some more examples of TONYPLOT command line options, with descriptions of what they mean:

1. To load two SPISCES log files `temp340.log` and `temp450.log` and display the graphs overlayed in a single plot:

```
% tonyplot -overlay temp340.log temp450.log
```
2. To load a SSUPREM4 structure file called `meshX.str` and set its display to a previous set up stored in `mx.set`, and then load a file containing IV data in user data format:

```
% tonyplot meshX.str -set mx.set iv.data
```
3. To make a printout of a structure created in DEVEDIT and send that printout to a printer:

```
% tonyplot devedit.str -print devedit.ps  
% lpr devedit.ps
```

7.3: The Base Window

7.3.1: Overview

The base window of TONYPLOT contains the area where all plots are displayed. When there are no plots to display, i.e., when no files have been loaded, a banner page is displayed.

Along the top of the base window is the main menu bar, holding the basic control menus for TONYPLOT. Each of these menus is explained further below, and in some cases in greater detail later on in this manual.

On-line help is accessed by selecting the **TonyPlot Help** option from the **Help** menu.

7.3.2: File Menu

This menu allows access to file control operations as described below. These operations consist of; loading structures, set file control, a command stream, and exiting from TONYPLOT.

- **Load structure** creates a popup (see Figure 7-3) which can be used to load structures. A list shows the current contents of the directory specified at the top of the popup. All subdirectories are shown (as folder icons), as are all files matching the specified filter.

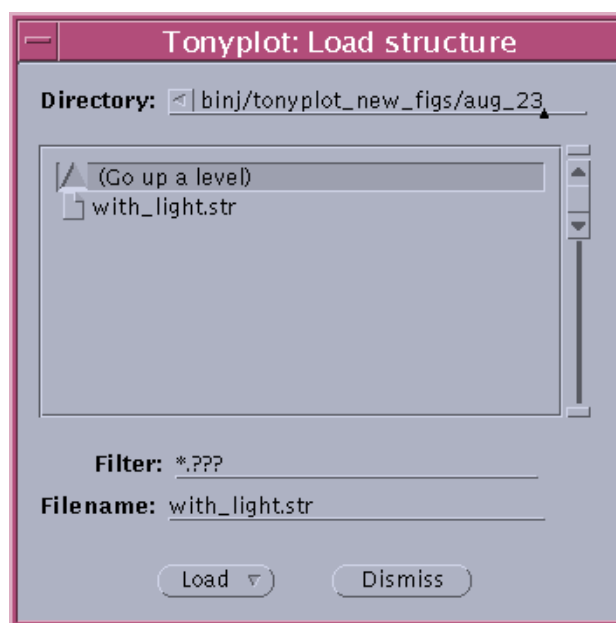


Figure 7-3: Load Structure Popup

- The first item in the list is **Go up a level**, which moves to the parent of the current directory. The directory can be changed by clicking on any line with a folder icon (including **Go up a level**), or by typing in a new path in the **Directory:** text field. To load a file, click on the name of the file in the list and choose an option from the **Load** menu. The options control the way in which the file is loaded: **Add** creates a new plot for the structure (this is the most common option and can be chosen simply by clicking on **Load** button). **Replace** replaces the structure(s) in the first selected plot in the view with the new structure. **Overlay** overlays the new structure on to existing structures in the first selected plot. If **Replace** or **Overlay** is chosen, and the view contains no selected plot, the effect is the same as choosing **Add**. If the desired file is not in the list, a name can be entered into the **File name** text field. Then choosing a **Load** option loads that file, if it exists.
- **Filter:** is used for screening irrelevant files and defaults to *.str to show structure files only. This can be changed in the **Filter:** text field if needed. When the Return key is pressed, the list is updated

to show files not masked by the new filter. The default file filter can be set from the **Miscellaneous** category of the **Properties** popup, described later.

- Clicking on the MENU button while the pointer is over the list of files displays the **File Loader Options** menu. This menu has options that allow directories and hidden ('dot') files to be shown or not shown, and also provides three criteria for sorting the files in the list: by name (alphabetically), by date (newest files first), and by size (largest files first).
- **Set files:** **Set files** (Figure 7-4) are used to save the display settings of currently loaded plots. This popup allows set files to be both created and loaded. It is very similar in appearance and operation to the **Load Structure** popup, except there are no **Load** options, although there is a **Save** button. To create a set file, enter a name into the **File name:** text field, then click on the **Save** button. To load a setfile, use the same methods as described for the **Load Structure** popup. Double clicking also loads a set file. Refer to the Set Files section for a more detailed description.

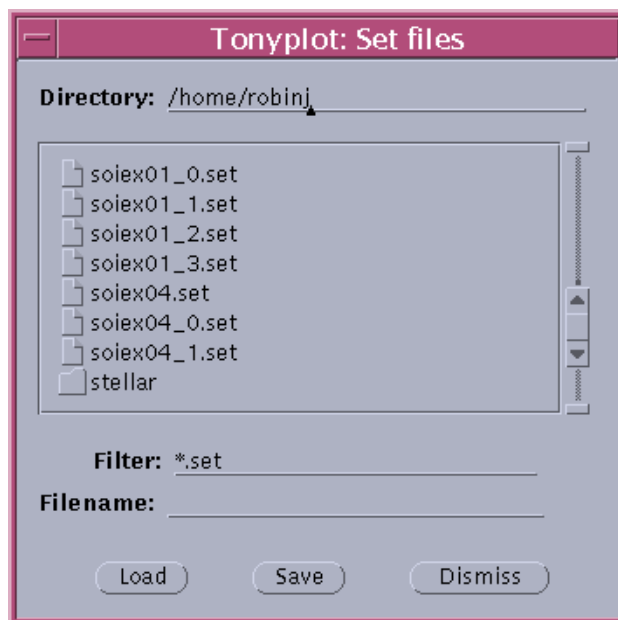


Figure 7-4: Set Files Popup

- **Export:** a feature that allows data files to be created from plots. To export data in this fashion, choose one plot that you wish to write to a file, and select this menu option. The **Export** popup appears. It is possible to export some files as **Master** files (Silvaco format) or in user-data format. Select the option desired. To name the file to be created, enter a base name and a file extension. One file is created for each overlay level in the plot. For example, if a three level plot is exported, and the basename is "hello" and the extension is "out", the following three files will be created:

```
hello_01.out
hello_02.out
hello_03.out
```

Some types of conversion are not allowed. If such a situation arises, TONYPLOT warns you that the specified export cannot be performed.

Production Mode: When used in conjunction with VWF, TONYPLOT provides very powerful **Production Mode** capabilities. This can be enable by selecting this option. Once enabled, these features remain active. The **Production Mode** can also be activated by using the **-production** option flag when starting TONYPLOT from the Unix command line.

Quit TonyPlot: Choosing this item removes all plots, structures and subwindows, and then quits TONYPLOT. Just to be sure, TONYPLOT asks to confirm this action first.

7.3.3: View Menu

The **View** menu provides control over the current view using operations that affect groups of plots as described below. On occasion, items on this menu become inactive (grayed out). This occurs when the operation is not applicable to the current group of selected plots.

- **Redraw all plots:** If at any time the view needs to be updated, and all windows need refreshing, choose Redraw all plots from the **View** menu. This item is always active.
- **Select all plots:** This item provides an easy way to apply an operation to all plots in the view. All plots become selected regardless of their previous state. This item is only active when one or more plots are loaded.
- **Swap two plots:** This item permits the ordering of plots in the view to be changed by swapping the positions of any two selected plots. It is most useful in the Palette Display Mode (see the Properties section).
- **Make overlay:** Overlay plots are comprised of several structures drawn in one subwindow. They are created by selecting a group of plots in the current view, and then choosing this menu item. A new plot is created with the structures overlayed. One of TONYPLOT's properties controls whether the original plots remain in tact or are deleted. This item is only active when at least two plots are selected. Refer to the Overlay section for a more detailed description.
- **Split overlay:** This option breaks an overlay plot up into separate single-level plots. This item is only active when at least one overlay plot is selected.
- **Previous Page/Next Page:** When the **Main Window** layout is set to **Page** mode (see Properties), then these two options become available. They cause the currently displayed plot page to be replaced with the previous or next page, respectively, wrapping around at the first and last page.
- **Duplicate selected:** All selected plots are duplicated if this is chosen. For each one, a new window is created, and the structure is drawn in the new window as well as the original window.
- **Delete selected:** Plots can be removed from the view with this item. All selected plots are deleted. This item is only active when at least one plot is selected.

7.3.4: Plot Menu

The **Plot** menu provides control over selected plots. See Section 7.19.2: "Plot Options" for a description of the menu items. The items on the menu can operate on many selected plots of the same type. If plots of different types are selected, only those with the same type as the first selected plot are affected.

7.3.5: Tools Menu

Many of the advanced features of TONYPLOT can be accessed from the **Tools** menu. On occasion, some of the items on this menu may become inactive. This indicates that there is no selected plot to which that tool can be applied. Each tool discussed below is described in detail in the **Tools** section. The following descriptions provide a brief summary of their uses.

- **Cutline:** Cross section profiles may be made from two dimensional data sets. Using the **Cutline** tool generates one dimensional cross section profile plots. Also available with the **Cutline** tool is the ability to move cutlines through a device and watch the profile move and shift as the cutline position updates. The **Cutline** tool also allows these sequences to be combined into a **Movie**. This item is only active when at least one 2D-Mesh plot is selected.
- **Ruler:** Coordinate information can be obtained from the **Ruler** tool. The **Ruler** shows useful data such as length, gradient, and intercepts, of a line you defined. This item is only active when at least one plot is selected.
- **Probe:** The **Probe** tool is used to examine a structure at spot locations. It provides both geometry and impurity data, and can also be used to find specified structural features. This item is only active when at least one 2D-Mesh plot is selected.

- **Movie:** The **Movie** tool combines a group of plots into an animated sequence and provides a collection of playback controls such as speed, direction, frame advance, and so on. This item is only active when at least two plots are selected.
- **HP4145:** This tool provides TONYPLOT with HP4145 emulation capabilities. The controls are designed to resemble those on a standard HP4145, including their operation. This item is only active when a single **Graph** plot is selected.
- **Integrate:** One dimensional cross section profiles can be used with this Integration tool, which calculates areas under a profile between specified x limits. This item is only active when a single **Cross-Section** plot is selected.
- **Trace path:** When a structure contains vector information, tracers can be placed within the structure. These will follow the path of the vectors and can trace out flow lines.
- **Poisson Solver:** The built-in Poisson solver

7.3.6: Print Menu

The **Print** menu provides all the necessary control for creating hard copy output and print files from TONYPLOT. See Section 7.17: “Printing” for a full description of TONYPLOT printing capabilities. The menu consists of these four options:

- **Print view:** Prints the view according to the current settings.
- **Options:** Sets up the parameters used when printing the view.
- **Printers:** Displays the printer editor for changing the list of printers known to TONYPLOT.
- **Forms:** Displays the form editor for changing the page formats known to TONYPLOT.

7.3.7: The Production Menu

When TONYPLOT is used in conjunction with the VWF, and the **Production Mode** is enabled, this menu is available. It contains two groups of options: the first group is a list of all the main **Production Mode** features, and the second group allows direct access to some parameter editing popups used in **Production Mode**. See Production Mode for a full description of these advanced topics.

- **Interactive:** Allows interactive user control over regression model parameters, to study the effects on the response surface.
- **Failure Analysis:** Predicts the most likely cause of failure in a production situation, given the characteristics of the input parameters and the failed condition.
- **Disposition:** This **Production Mode** feature is not yet available.
- **Calibration:** This **Production Mode** feature is not yet available.
- **Synthesis:** After a set of outputs required from a production environment is given, this feature calculates the best set of inputs that should be used to achieve that goal.
- **Yield Analysis:** This can predict the characteristics of output yield from a production situation, from known experimental data of the following input parameters.

Each of the following options provide access to the various parameter editing popups that are used in **Production Mode**. These options are also available from the main Production popup as well.

- **Input parameter ranges**
- **Input distributions**
- **SPC limits**
- **Experimental results**
- **Optimizer setup**
- **ASA setup**

7.3.8: Properties Menu

You can alter many of the default actions, features and parameters in TONYPLOT to suit your preferences. The **Properties** menu shows a list of all the categories of properties that exist. For a full description see the Properties section.

7.3.9: Help Menu

The **Help** menu provides access to the online assistance available in TONYPLOT. There are four items in the **Help** menu.

- **TonyPlot Help:** Shows the latest user's manual for TONYPLOT in PDF format.
- **Release Notes:** Displays a help window containing information about the current release of the program.
- **Email:** Displays the popup that provides an interface to electronic mail. With this popup, you can compose a message that you wish to send to the support staff at Silvaco, and deliver it, without leaving TONYPLOT. For maximum benefit, it is strongly recommended that you complete all sections on the **Environment Properties** popup (described in a later section).
- **About TonyPlot:** Shows a popup notice displaying the version number of the program and its component libraries.

7.4: The File Loader

This popup can be used to load structures into TONYPLOT. A list shows the current contents of the directory specified at the top of the popup. All subdirectories are shown (as folder icons), as are all files matching the specified filter.

7.4.1: Loading files

To load a file from the File Loader, highlight the name of the desired file in the scrolling list (by clicking on it) and then click on the **Load** button. This creates another plot inside TONYPLOT to display that data. To select multiple files, hold shift while highlighting filenames. Alternatively, enter the name of the file to be loaded into the field marked **Filename**, and then click on the **Load** button. The **Load** button actually has a menu attached to it, and this provides three different load options:

- **Add:** the default load method, it adds the new plot to the current TONYPLOT window.
- **Replace:** Replaces the first selected plot with the new plot that is created.
- **Overlay:** Overlays the data on to the data already in an existing plot, if there is a plot selected. The data is loaded and overlayed on to the first selected plot of the same type.

7.4.2: Changing Directories

The directory that TONYPLOT is currently looking at is shown at the top of the **File Loader**, and its contents are shown in the scrolling list below. The first item in the list of files is always **Go up a level**, which moves to the parent of the current directory. Double-clicking on this line moves up a level. To change to a subdirectory, double-click on its name, and TONYPLOT moves down a level.

It is also possible to type in a path name into the text field labeled **Path** and press the Return key, to immediately change the directory.

7.4.3: File Filtering

A filter is used for screening irrelevant files and defaults to *.??? to show the types of files normally produced by Silvaco simulators and other tools, such as the VWF. This filter can be changed in the **Filter:** text field if needed. When the Return key is pressed, the list is updated to show files not masked by the new filter. The default file filter can be set from the **Miscellaneous** category of the **Properties** popup, described later.

Note: Loading files by entering the name into the Filename: field will still load the file even if the filter prevents it from showing up in the scrolling list.

7.4.4: File Options

Clicking on the MENU button while the pointer is over the list of files displays the **File Loader Options** menu. This menu has options that allow directories and hidden (“dot”) files to be shown or not shown, and also provides three criteria for sorting the files in the list: by name (alphabetical), by date (newest files first), and by size (largest files first).

7.5: Plot Control

Each plot has an associated group of parameters, collectively known as a display setting, which determine how the plot is to be drawn in its subwindow. These parameters are independent from the structure(s) that are represented in the plot, so can be applied to a number of other structures to make data comparison easier. A plot is controlled using a combination of the following techniques:

- Plot Selection
- Pointer Zooming
- Key Commands
- Command Stream
- Plot Menu

Some plots can be displayed in a 3D mode. These plots have slightly different rules for control. Zooming is not available, but it is possible to rotate and scale 3D plots.

7.5.1: Plot Selection

In order for plots to be controlled and their display parameters changed, one or more must be selected. When the view consists of only one plot, it is always selected. When the views consists of multiple plots, you must select those to be used. Selection is achieved by using either the mouse **SELECT** button (to select just one plot), or **ADJUST** button (to toggle the selection state of the plot). Also, choosing the **Select all plots** item on the **View** menu automatically sets all plots to a selected state.

Plots that are currently selected have a highlighted border, while the border on unselected plots remain normal. The defaults for these colors are white and gray, but can be changed from the **General Colors** category of the **Properties** popup.

If the main window layout mode is set to **Page**, then only the plot on the current page is selected, all plots on the other pages are unselected.

7.5.2: Pointer Zooming

To zoom in on a specific portion of a plot, a method known as pointer zooming can be used. By dragging the pointer across the plot, a dynamic box can be drawn around the area of interest. If the SHIFT key is held down while the mouse is being dragged, the start point of the rectangle can also be moved. When the mouse button is released, the plot is redrawn so that the area within the box fills the whole subwindow.

Note: All selected plots of the same type are zoomed in by the same relative coordinate of the zoom rectangle.

When at least one plot has been zoomed, the **zoom panner** appears (Figure 7-5). The panner consists of nine buttons; eight directional and a central **zoom out** button. Press the directional buttons to pan around the plot at the same zoom scale. Press the central diamond to zoom out, and restore the plots to their original sizes.

Note: Zooming can also be done by specifying the exact coordinates of a zoom rectangle. This is done using the Zoom popup, accessed by choosing **Set Zoom...** from the Plot menu.



Figure 7-5: Zoom Panner

7.6: Key Commands

Key commands are available for some plot control. Point at a plot with the mouse and press one of the following keys:

- **d** (duplicate) duplicates the plot, as if **Duplicate** were chosen from the **View** menu.
- **g** (grid toggle) turns the axis grid on or off.
- **h** (help) prints out a list of these key commands on the standard output.
- **j** (junk data) prints a list of all “junk data” attached to the plot structures.
- **m** (mean & standard deviation) prints info about statistical data column (statistics plots only).
- **n** (next page) changes to the next page when in **Page** mode.
- **N** (previous page) changes to the previous page when in **Page** mode.
- **p** (position) displays the current pointer x-y position in a popup window.
- **r** (region data) prints all attached region data for all structures in the plot.
- **v** (value label) adds a spot height value label to contour plot (RSM 2D plots only)
- **z** (previous zoom) sets the zoom level to the zoom level previously used.
- **Z** (autozoom) zooms all mesh plots to the quadrant containing the greatest number of mesh points.
- **ESC** (abort drawing) aborts contour drawing at any point. This only works if the TONYPLOT window has “keyboard focus” when drawing begins.
- **b** (bias) display bias information for structure from device simulation.

7.7: Command Stream

As well as the normal graphical user interface control to TONYPLOT, a command language known as a **Command Stream** is available.

To start a command stream, choose **Command Stream** from the main **File** menu. A prompt appears in the window from which TONYPLOT was started. Commands can be entered at this prompt, and the effects seen in the popups and windows of TONYPLOT.

The syntax of the command language is tightly linked to the popups and their controls. Each statement mimics an item on a popup or an input action. When using a command stream, you should be aware that:

- Plots must be selected when using the commands stream just as they do when using the normal GUI. The **Select** command is used to do this.
- Changes are never seen until a **Redraw** statement is given. This is like clicking on an **Apply** button.
- The command stream can be used at the same time as the normal GUI, but only one command stream can be started at one time.
- The contents of “set files” are lines of command stream syntax. Set files are therefore good examples of command stream syntax. Also, set files can be constructed “offline” using command stream syntax.
- When starting TONYPLOT from `csh`, TONYPLOT must be the current foreground process for the command stream to start. If TONYPLOT is not the foreground process, it “stops” until you make it the foreground process. This does not apply to loading set files.
- To finish a command stream, type **Control-D**.

7.8: The Plot Menu

The **Plot** menu can be accessed either by clicking on the **MENU** button while the pointer is over a selected plot, or by using the set of controls on the main frame. When accessed with the MENU button, the plot menu (Figure 7-6) has a title that indicates the type of plot under the pointer.

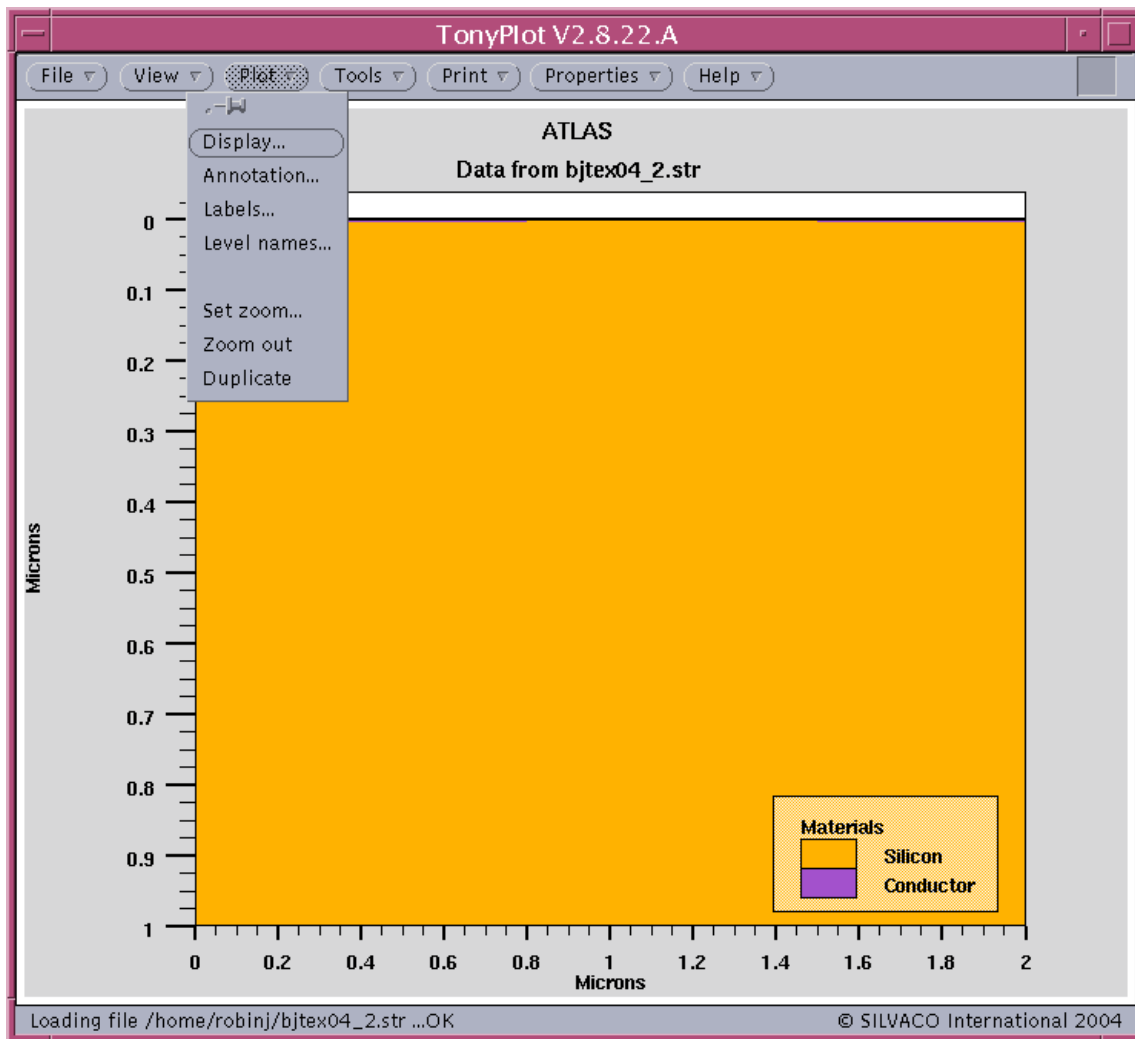


Figure 7-6: The Plot Menu

The Plot menu contains the following items:

- **Display:** accesses the **Display** popup for the plot. This popup is used to change **Display Parameters**, i.e., how the data is displayed. For full details, refer to the Display section.
- **Annotation:** displays the **Annotation** popup, which allows control over titles, axes, and so on. Refer to the Annotation section for more information.
- **Labels:** accesses the **Labels** popup. This is used to add notes and labels to a plot as titles, markers, etc. Refer to the Labels section for more details.
- **Level names:** allows you to change the names assigned to overlay plots. Overlay plots use the data file name as a default name for each level. This popup also allows you to select whether graph lines have points or lines for individual levels. Refer to General Information section for details about overlays.

- **Set zoom...:** displays the popup shown in Figure 7-7. This popup is used to define a data range to which the plot will be zoomed. Enter the opposite corners of a rectangle, and click on the **Apply** button to set this zoom. Any previous zoom parameters will be discarded.

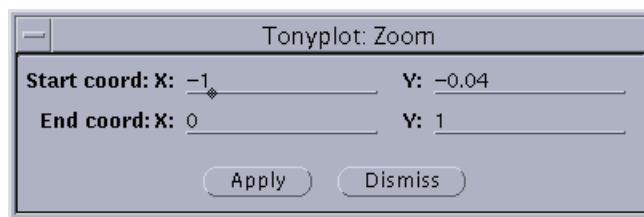


Figure 7-7: The Zoom Popup

- **Zoom out:** restores the plot to its normal size.
- **Duplicate:** creates a new subwindow and plot, and copies the data from this plot into the new one. The new plot displays the same data, but can be controlled separately.

7.9: 3D Plot Control

3D plots can be rotated and scaled, but cannot be zoomed.

7.9.1: Rotation

To rotate a 3D plot, hold down the left mouse button and drag the pointer left and right over the plot. A wire-frame bounding cube will be drawn around the plot and rotates as the mouse is moved. Position this cube to the desired viewing angle, and release the mouse button. The plot is redrawn from the new view point.

7.9.2: Scaling

To scale a 3D plot, hold down the Shift key and the left mouse button, and drag the pointer up and down over the plot. A wire-frame bounding cube is drawn around the plot and grows or shrinks as the mouse is moved. Scale this cube to the desired size, and release the mouse button. The plot is redrawn at the new size.

7.10: Plot Display

Choosing the way data is displayed in a plot is the heart of TONYPLOT. Much information can be shown in a plot, and choosing the correct subset of this information is essential if a useful plot is to be drawn. There are three distinct types of plot, each with its own set of display parameters. These are 2D Mesh plots, XY Graph plots, and Cross Section plots. The method used to set up the display for each of these is described in the following sections.

7.10.1: 2D Mesh Plot Display

When the **2D Mesh Plot** popup appears (Figure 7-8), it shows the current display settings for the first selected **2D Mesh** plot. When the settings on the display popup are applied, all selected 2D Mesh plots are affected. In this way it is easy to apply global changes to similar plots in the view. The popup shows the 9 features that can be displayed in 2D Mesh plots.

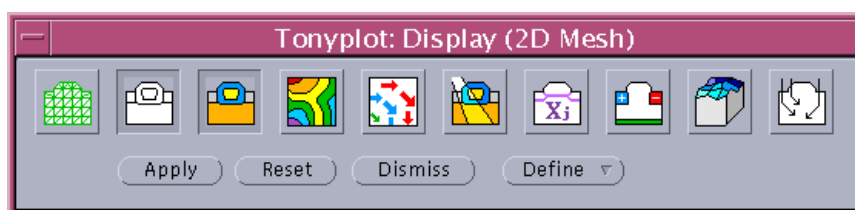


Figure 7-8: 2-D Mesh Plot Popup

The icons shown above symbolize (from left to right):

- **Mesh:** The triangular mesh used in the simulation.
- **Edges:** Sides of triangles classified as region edges.
- **Regions:** All material regions in the structure.
- **Contours:** Color plotting of impurity values.
- **Vectors:** Representation of vectored impurities.
- **Light:** Light beam and ray information.
- **Junctions:** Metallurgical junctions in the semiconductor regions.
- **Electrodes:** Regions defined as being electrodes
- **3D:** Adds elevation to a plot so that 3D surface is plotted
- **Lines:** Adds lines dates to plots for ionization integrals or Monte Carlo ion implant.

Some of these features have further control popups, and these can be accessed from the **Define** menu. Clicking on the **Define** button shows popups for all features that have been chosen, while choosing an item from the menu shows just that corresponding popup. The features that have detailed control are: Regions, Contours, Vectors, Light and 3D.

Regions

There are a number of ways that TONYPLOT can display mesh regions. These are available from the **Regions** popup (Figure 7-9), which is accessed from the **Define** menu.

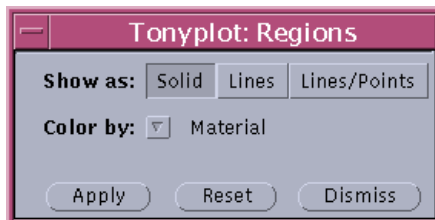


Figure 7-9: Regions Popup

The first choice controls the way in which regions are drawn. The options are **Solid**, which fills the region area with color, **Lines**, which draws the region outlines only, and **Lines/Points** which draws the region outlines and marks the points defining the region border.

The second choice determines the parameter used when determining the regions color. The first two options are always available: Material and Region. If “Material” is chosen, the color represents the material of the region (silicon, oxide, etc.) and the region key shows the material names. If “Region” is chosen, then each region has its own unique color, and the key shows the name of each region. If the data contains further region information (i.e., workfunction, phase, etc), then these are also available, and the key shows the values of these parameters in each region. Any regions that are one-dimensional (i.e., substrate electrodes) are drawn as thick lines, since they do not enclose a complete area.

Contours

Contouring is the most commonly used method for visualizing data on two dimensional meshes. The contouring facilities in TONYPLOT provide sufficient control for obtaining any desired plot. Both contour plots and fringe plots (filled contours) are available, with material naming and range control to limit the plot to a restricted subset of the data. Each plot can have up to three sets of contours displayed at once. This makes it possible to view more than one quantity simultaneously, either all filled but in different material regions, or all lines over all materials, or any other combination. Of course, if all three sets are filled sets and all are plotted over the same materials, only the third set (the last one to be drawn) will be visible. If lines and filled sets are combined, the filled set should come before the line set.

TONYPLOT selects a default quantity whenever possible. This allows contours to be plotted without the need to use the **Contours** popup (Figure 7-10).

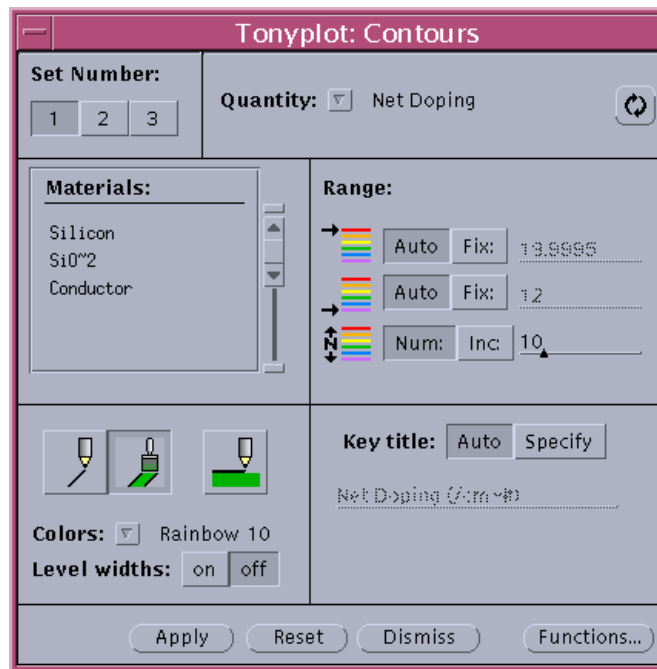


Figure 7-10: Contours Popup

Just select contours from the **Mesh Display** popup, and contours are plotted when the **Apply** button is clicked on. The **Contours** popup appears if contours are selected on the **2D Mesh** popup and the **Contours...** button is clicked on. The popup is divided into subsections, as follows:

- **Set number:** This subsection shows which set is currently being edited. Set 2 will only be plotted if set 1 is plotted, and set 3 will only be plotted if set 2 is plotted. For a set to be plotted, the quantity (see below) must be anything other than **None**.
- **Quantity:** Choose the quantity to be contoured. If the range items (see below) are set to Auto, the corresponding minimum and maximum text fields are updated to show the range of the new quantity. If None is selected, the current contour set is not be plotted. One of two functions may also be chosen. Functions are defined from the **Functions** popup.
- **Materials:** The part of the structure on which contours are drawn can be limited to regions of a certain material. If no materials are selected in the list, this is treated as all being selected (the default). If the contours should not be plotted on any material, set the Quantity to **None** (see above).
- **Range:** The group of items on the right part of the popup control the range of the data through which contours are plotted. The maximum (top item), minimum (middle item), and interval or number of steps (bottom item) can be set or left to automatic defaults. The default minimum and maximum values are the minimum and maximum values of the data in the structure(s). The default number of steps is the same as the number of colors in the current color set (see below).
- **Drawing style:** Selecting the pencil creates line contours, while the paintbrush creates filled contours (fringe plot). If filled contours are plotted, optional outlines can be added by selecting the Outline icon, which is to the right of the paintbrush. TONYPLOT provides several color sets which can be used for contour plotting. If the contour range is determined by the number of steps (Num: selected), the number updates to match the number of colors available in the color set when one is selected.
- **Level widths:** If the drawing style is set to line contour, this option forces overlaid plots to use a different line width for each level.
- **Key title:** The key title can either be set automatically by TONYPLOT (set to Auto), or entered by hand to create a custom title (set to Specify:). An automatic title consists of the name of the quantity

plotted, with units if appropriate. A custom title is created by entering the desired text into the text field supplied. This title is used on the contour key for this contour set.

- **Functions** — Click on this button to display the Functions popup, which can be used to define the functions of the original quantities that can be selected from the choice of Quantities.

Vectors

Vectors can be plotted for standard (the default) or user-defined vectorial quantities. TONYPLOT automatically detects the standard quantities made of an X component and a Y component. They are shown in the Quantity pulldown menu. To create a vector made of unrelated X and Y quantities, select the **Custom** option.

Vectors are represented on the plot by arrows. The direction of the arrow shows orientation of the vector. The color or length of the arrow or both shows the magnitude of the vector. The Vectors popup (Figure 7-11) will appear if you choose **Vectors** on the Mesh 2D display popup and if you select the **Define→Vectors....**

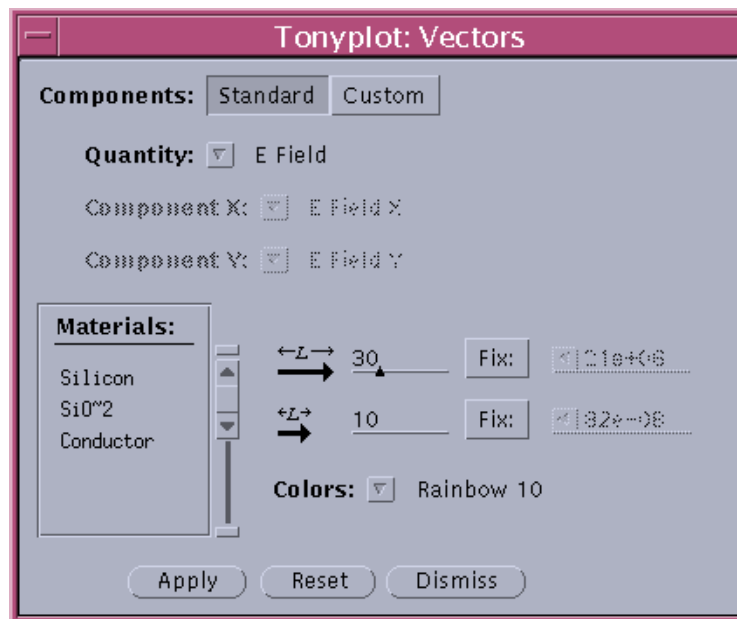


Figure 7-11: Vectors Popup

The items on the popup are as follows:

- **Components:** This option controls whether to draw standard or user-defined vectorial quantities.
- **Quantity:** All quantities of the structure that are standard vectorial quantities are shown in this pulldown menu. This menu is only active when the **Components** option is set to **Standard**. The list of quantities includes the two functions defined in the Functions popup. See Section 7.19.15: “Functions” for more information.
- **Component X:** This menu is active only when the **Components** option is set to **Custom**. It is used to assign a quantity to the X component of the user-defined vector.
- **Component Y:** This menu is active only when the **Components** option is set to **Custom**. It is used to assign a quantity to the Y component of the user-defined vector.
- **Materials:** The part of the structure on which vectors are drawn can be limited to regions of a certain material. If no materials are selected in the list, then it is treated as all being selected (the default). If you do not plot the vectors in any materials, set the **Quantity** or **Component X/Y** to **None**.

- **Range:** The range items on the right hand side of the popup control the sizes of the vector arrows drawn. The longest arrow matches the vector with the greatest magnitude, and the shortest arrow matches the smallest magnitude. If the smaller length is specified as zero, then the lengths of the draw arrow are directly proportional to the vector magnitude.
- **Colors:** Specifies the color sets for the arrows. These are the same color sets that are used on the Contours popup.
- **Functions:** Displays the Functions popup used to define the two functions you can select in the Quantity pulldown menu.

Light

If a structure contains light ray information, the **Light** popup can be used to display that data in a number of ways. Light information consists of a number of Beams, and each beam is comprised of a series of Rays. A ray is a section of a beam between reflections and refractions. For example, if a beam originates from outside a structure, enters the structure (is refracted), travels to the bottom of the structure (gets reflected), and then moves back to the top of the structure, it consists of three rays. The **Light** popup (Figure 7-12) appears if Light is chosen on the **2D Mesh** plot popup and the **Light** button is selected in the **Define...** menu.

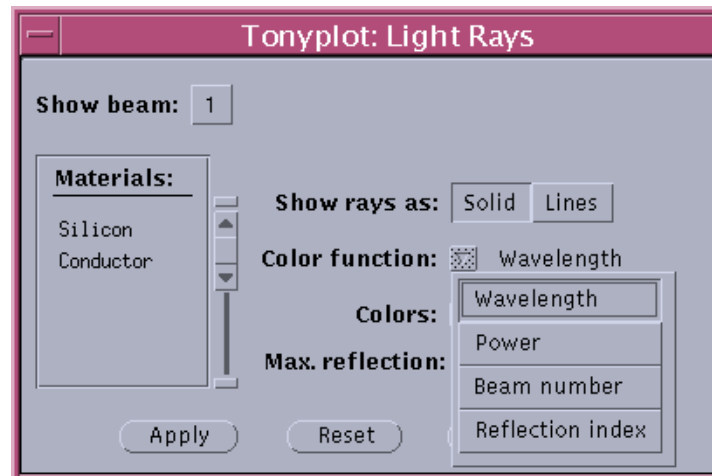


Figure 7-12: Light Rays Popup

The items on the popup are:

- **Beam:** When beams are present in the structure, they are shown as selectable numbers on this item. Any of the beams can be shown at once by selected the required beam numbers. If no light information is present in the structure, the option None is shown to show that there are no beams.
- **Materials:** The part of the structure on which light beams are drawn can be limited to regions of a certain material. If no materials are selected in the list, this is treated as all being selected (the default). If the beams should not be plotted on any material, deselect all beam numbers.
- **Show as:** There are two ways to show light beams: either as lines that show the path of the beam or solid areas that also show the width of the beam. Choose the required option with this item.
- **Color function:** The light beam rays can be colored in a variety of ways: **Wavelength** colors the rays to match the wavelength. Wavelengths less than ultra-violet are shown as magenta, and those above infra-red as pink. **Power** assigns a color from the chosen color set dependent on the beam intensity. **Beam number** assigns one color to each beam. **Reflection Index** assigns the color of a ray according to the number of times it has been reflected.
- **Colors:** Allows a color set to be chosen. These are the same as those available on the **Contour** and **Vector** popups.

- **Maximum reflection:** This can be used to limit the number of rays drawn. Only rays that have been reflected a number of times equal to or less than the number shown are drawn.

Junction

You can plot depletion region edges from device simulations (Figure 7-13), as well as metallurgical junctions. Depletion factor is the value of the ratio of majority carriers to doping used to determine the depletion region edge.

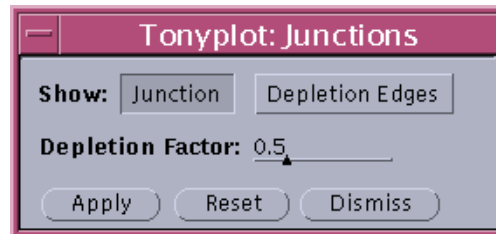


Figure 7-13: Junctions Popup

3D

A structure that can be contoured can also be elevated, by choosing the 3D option and defining some parameters from this popup.

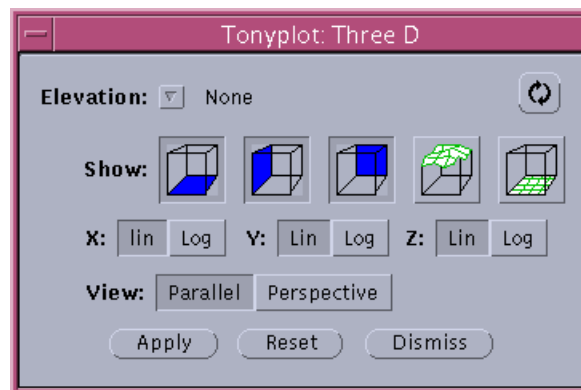


Figure 7-14: 3D Popup

Elevation: Any quantity can be chosen as the elevation. The height of the surface at any point is proportional to the value of the elevation quantity.

Show: There are number of optional items that can be drawn on a 3D plot, and each is described by a small icon on the “Show” item. Choose the ones desired from this list.

View: The view projection can be one of two choices: “parallel” or “perspective”: choose the one desired.

Log: You can select whether to log each of the axes individually.

Lines

It is possible to overlay lines onto a 2D plot (Figure 7-15). This is used to show static field lines on ion implant tracks.

Color: Lines can be plotted in the same single color or be multi-colored.

Number: Select number of lines to appear on the key.

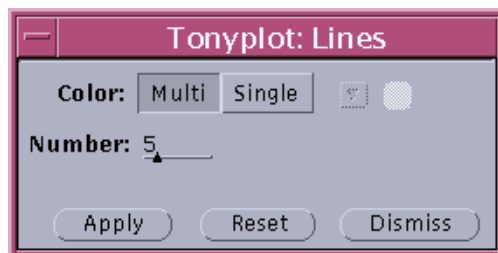


Figure 7-15: Lines Popup

7.11: X-Y Graph Display

The **Graph Plot** popup (see Figure 7-16) shows the current display settings for the first selected graph plot. When the settings on the display popup are applied, all selected graph plots are affected. This permits much easier application of global changes to similar plots in the view. The **Graph Plot** popup contains all the controls needed for complete control of graph plots.

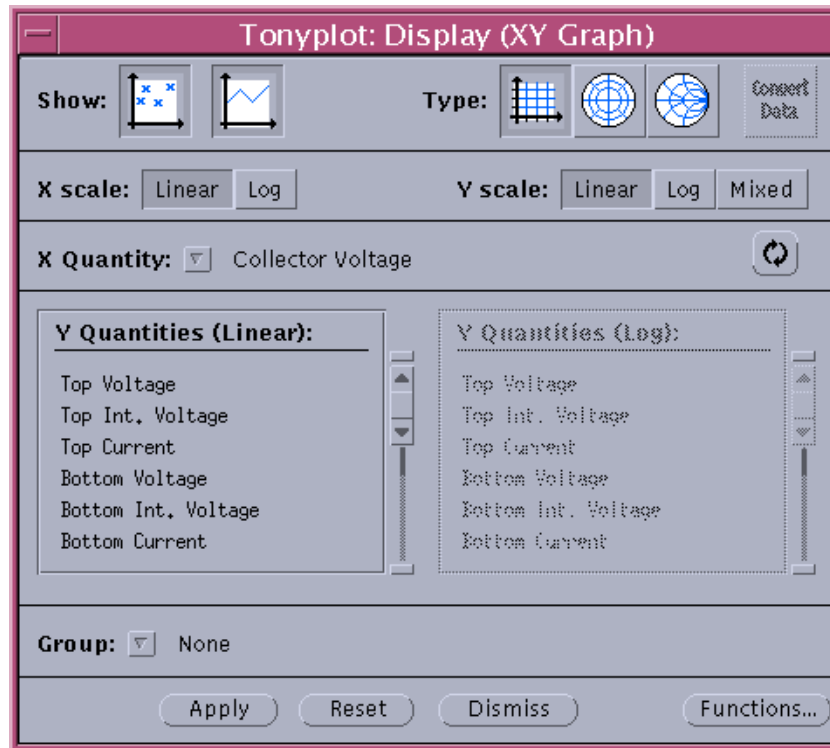


Figure 7-16: Graph Plot Popup

Show: Allows selection of the way lines are drawn on the graph. Points and/or lines can be chosen.

Type: Specifies the type of graph that is to be drawn. The options represent cartesian graphs, polar plots, and Smith charts. The data that is plotted is mapped on to axes of the chosen type. If the data is in a format that matches the type chosen (i.e., [r, theta] pairs for polar plots), then selecting the **Convert Data** button tells TONYPLOT to transform the coordinates before plotting them.

Functions: Displays the **Functions** popup, which can be used to define the functions that can be selected from the choice of **Quantities**. Functions can be plotted on any axis. The type of graph chosen will affect the controls on the remainder of the **XY Graph Display** popup.

7.11.1: Cartesian Graphs

Scales

Since only one quantity can be plotted on the x-axis, there is an item to select a linear or log (base 10) x-axis. Choose the one desired. For the y- axis, more than one quantity can be plotted. If all of them are to be on a linear scale, choose **Linear** for the y scale. Choose **Log** if they are all to be log, and **Mixed** if both linear and log quantities are to be plotted on the y-axis.

X Quantity

One quantity can be chosen for plotting on the x-axis. All quantities available in all selected plots appear in this list. If one of the plots does not have data for the chosen quantity, nothing is drawn.

Y Quantities

Any number of quantities can be chosen for plotting on the y-axis. All quantities available in all selected plots appear in two lists. There is a list for choosing quantities to be plotted on a linear axis, and a list for those to be plotted on a Log axis. Which lists are active depends on the setting of the “Y scale” item above.

A menu attached to each list (accessed by pointing to the list and clicking on the MENU mouse button) makes list control a little easier. There are options to move selections from one list to the other, and for rapidly selecting, deselecting and locating choices in the lists.

Group

When Cartesian data is plotted that contains different groups of data sets for the same y quantity, this item can be used to specify which quantity divides the y value into its distinct groups. For example, a structure may contain data to show several plots of drain current against drain voltage for different values of gate voltage. In this case, the x-axis would be set to “drain voltage”, the y-axis to “drain current” and the Group item to “gate voltage”. The plot would show one curve of I_d vs. V_d for each value of V_g .

7.11.2: Polar Charts

When the graph type is Polar, four subpanels are shown. Each subpanel can be used to display quantities, i.e., up to four polar curves can be plotted.

Two quantities are used to specify each curve. By default, the quantities real and imaginary are used when the data is not converted. If the data is to be converted, the quantities R (radius) and A (angle) are used. The data should only be converted if it appears in (r, theta) form in the structure.

When quantities are present that TONYPLOT recognizes as being usually displayed on polar charts, TONYPLOT tries to automatically select an “i” (or “A”) quantity whenever you choose an “r” (or “R”) quantity.

The real or radius quantity can be logged before plotting, and the angle quantity can be specified in terms of degrees or radians. Choose the setting which corresponds to the data in the structure.

There are some options to control the polar chart drawn. The chart can be drawn proportionally (i.e., concentric circles appear as circles, even if the plot window is not square), and radial labels can be shown in degrees (radians is the default). The radial lines can be drawn at various intervals; choose the interval desired from the item marked “Radials:”.

7.11.3: Smith Charts

When the graph type is Smith, four subpanels are shown. Each subpanel can be used to display quantities, i.e., up to four Smith curves can be plotted. This is basically the same as Polar charts, described previously.

Two quantities are used to specify each curve. By default, the quantities real and imaginary are used when the data is not converted. If the data is to be converted, the quantities R and X are used. The data should only be converted if it appears in R, X form in the structure.

When quantities are present that TONYPLOT recognizes as being usually displayed on smith charts, TONYPLOT tries to automatically select an “i” (or “X”) quantity whenever you choose an “r” (or “R”) quantity.

There are some options to control the Smith chart drawn. The chart can be drawn proportionally (i.e., concentric circles appear as circles, even if the plot window is not square), and axis arms can be drawn in all four Smith quadrants (only the first quadrant is shown by default).

7.11.4: Cross Section Display

The **Cross Section** popup (Figure 7-17) shows the current display settings for the first selected **XSection** plot (see the Cutline section for details on how to generate **Xsection** plots). When the settings on the display popup are applied, all selected **XSection** plots are affected. This permits much easier application of global changes to similar plots in the view. The **Cross Section** popup contains all the controls needed for complete control of **XSection** plots.

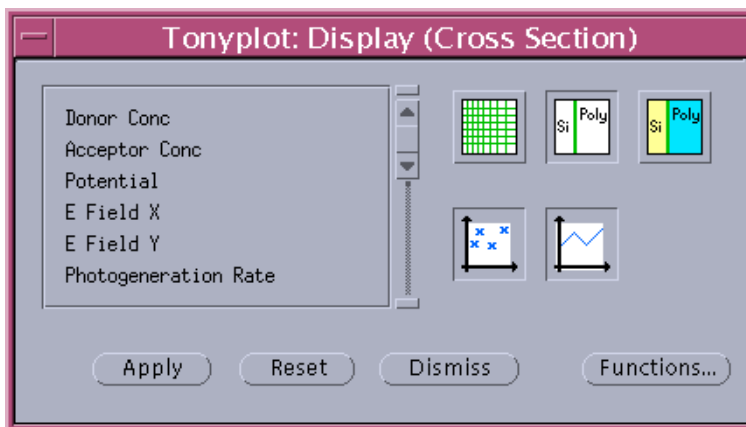


Figure 7-17: Cross Section Popup

The following items may be specified:

- **Quantity:** Specifies quantities to be plotted. The available quantities are listed in the list that appears on the left hand side of the **Cross Section** popup window. Any number of quantities can be plotted. One of two functions may also be chosen. These functions are defined from the Functions popup. Refer to the Functions section for more detailed information.
- **Options:** Allows addition of mesh, interfaces, and/or materials to the plot. These options are represented by the three icons in the top right of the popup. To add any of these features to the plot, select the corresponding icon. The icons underneath control the way lines are drawn on the graph. The icons allow points and/or line segments to be drawn).
- **Axis Scales:** Each quantity that can be plotted on a Cross Section plot has a default flag which TONYPLOT uses to determine whether a Linear or Log y-axis should be used. If linear and log quantities are plotted simultaneously, two y-axes are drawn, one to show all log quantities, one to show all linear. These internal flags can be changed, using the menu attached to the list: point to the list and click on the MENU button to display this menu.
- **Functions:** Click on this button to display the **Functions** popup, which can be used to define the functions that can be selected from the choice of Quantities. Refer to the Functions section for more detailed information.

The type of y axis drawn depends on the quantities being plotted. If log scale quantities are drawn, such as Net Doping, then TONYPLOT draws a true log scale axis. If linear quantities are drawn, such as Potential, a normal linear axis is drawn. If a mix of the types of quantities are drawn, TONYPLOT draws both types of axis: the log axis appears on the left side of the subwindow, and the linear axis on the right. When reading values from the curve, be sure to use the correct scale.

For dopants, the log axis shows values below 1e12. This value can be changed in the **Plot Options** panel of the **Properties** popup (see the Properties section).

7.12: RSM Display

The **RSM Plot** display popup allows control over the RSM (response surface model) inputs and outputs that are displayed, and how they are displayed. RSM plots can be drawn in one of three modes: 1D graphs, 2D contours, or 3D surfaces.

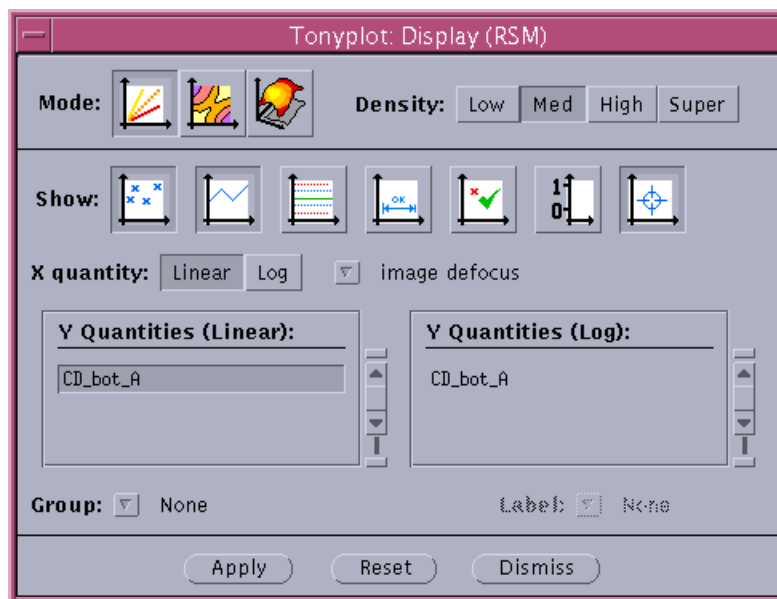


Figure 7-18: RSM Popup

To draw any RSM plot, TONYPLOT “samples” the input(s) a number of times to calculate values for the output. The number of samples taken (which are always regularly spaced) is determined by the setting of the **Density** item. Higher densities generate more points, creating smoother curves or surfaces, but take longer to compute. Low density plots are quick to calculate and draw, but provide only “approximate” plots.

For total control of RSM plots, and access to a selection of analysis tools, the VWF PRODUCTION MODE should be enabled. A description of these features can be found in the Production Mode section. Without **Production Mode**, the benefit of RSM plots is greatly reduced.

7.12.1: 1D RSM Graphs

When RSMs are plotted in the 1D mode, any one input can be selected for the x- axis, and any output(s) can be selected for the y-axis. All RSMs that contain both the input and output are plotted.

Show

There are a number of ways to display data on 1D RSM plots, and these are selected from the row of icons near the top of the control panel. **Points/Lines:** The first two icons draw points and line segments at or between sampled points, to draw the curve. **SPC** limits can be added to the plot, if this information is available for the outputs being plotted. The next icon activates the **Valid X Range** marker, showing the range of the input that is valid for the model being used (the drawn range can be extended in certain cases to values outside the valid range). The next icon represents **Measured Points** which are plotted if measured data was passed with the RSM. Next, the icon with the zero and one normalizes all outputs to a range between 0 and 1, for easier comparison of different models. The last icon activates a **Gunsight**, which can be used to track x- and y-coordinates along the curve.

X Quantity: Any input parameter can be chosen for plotting on the x-axis. In addition, values of this input can plotted on a log scale.

Y Quantities(s): Any output parameters can be chosen for plotting on the Y axis. There are two lists: one for outputs to be plotted on a linear axis, and one for a log axis. Any combination of the two can be used.

7.12.2: 2D RSM Contours

In the 2D mode, RSM plots show how outputs vary with respect to two independent input parameters. The way contours are drawn can be specified, in the same way contours are drawn in regular 2D Mesh plots from ATLAS or ATHENA.

X and Y Quantities

Two inputs must be chosen for contour plots, one for the x-axis and one for the y-axis. The inputs chosen must be different. Each of these can be plotted on a log scale if desired.

Z Quantity

The Z Quantity cannot be chosen (it is always the RSM output parameter that is plotted), but it is possible to specify a linear or log scale for the Z axis.

Contour Type

Contours can be drawn as lines or filled areas, and filled areas can be outlined. There are a number of color sets that can be used to create the contours. All these options are controlled with the items in the lower left corner of the control panel.

Mesh

Next to the icons for controlling the contour types is an icon that draws the “sample mesh” on top of the contour plot. This mesh shows the points where outputs were calculated in order to generate the plot. A triangular mesh is created from these points in order to draw contours.

Projection

This is not used in the 2D mode. See the description of the 3D mode for an explanation.

Output Range

The range over which contours drawn can be selected as one of two options. The first option is to use the highest and lowest output values over the sampled input range. This ensures all contour colors are drawn on the plot. The second option is to use the absolute range of the model output, which can be a greater range than that plotted.

7.12.3: 3D RSM Surfaces

When the 3D mode is chosen, TONYPLOT draws an RSM as a three dimensional surface, with contours drawn according to surface “height”. Control over the display of these plots is the same as described above for 2D plots, except that projection may also be specified. **Projection** allows you to choose either parallel or perspective projections when the 3D surface is displayed.

Note: These 3D plots can be rotated and scaled (but not zoomed). See the **Plot Control** section for details on 3D rotation and scaling.

7.13: Statistics Display

Whenever statistics plots are present, the **Statistics Plot Display** popup can be used to alter the way the data is displayed. These **Statistics Plots** are often generated from PRODUCTION MODE tools (see the Production Mode section).

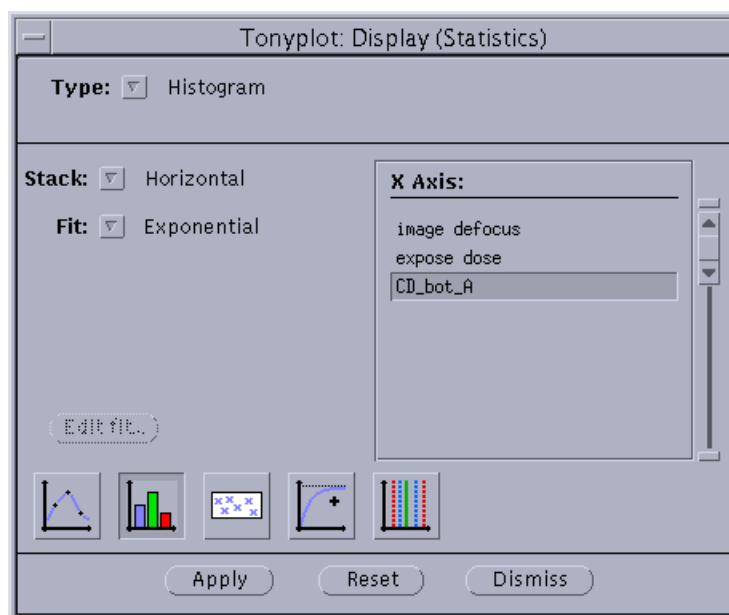


Figure 7-19: Statistics Popup

Statistics plots can be one of several distinct types; the current type is shown at the top of the popup, and can be changed to any other type. For each type, a different set of control appears beneath, on the lower portion of the popup. The types available are **Histogram**, **Pie Chart**, **Scatter Plot**, **Box Plot**, and **Sunray Plot**. Each is explained further in the following paragraphs.

Note: Some features of statistics plots are incomplete. There may be control items on the Display popup that are neither active, nor documented. These items control incomplete features, and show some functionality that will be available in a future update of TONYPLOT.

7.13.1: Histograms

X Axis

One or more quantities can be plotted on a histogram plot. Choose the quantities desired from the scrolling list to the right.

Stack

When more than one quantity is plotted, the stack item selects the method used for showing each one on the same axis: the bars for each quantity can be stacked vertically on top of each other, or horizontally next to each other.

Fit

This item allows a best-guess distribution curve to be plotted over the data. TONYPLOT uses the range, mean and standard deviation of the data to generate a distribution curve of the chosen type.

Show

The icon choices along the bottom represent various items that can be drawn on a histogram plot. These are Lines drawn from one bar to the next, **Solid Bars**, which are the default, **Jitter Plot** to show the distribution of all the data points, **Cumulative Curve** to show the total number of data points over the X axis range, and **SPC Limits**, which are drawn when RSM output quantities are plotted on the histogram. See the Production Mode section for information about SPC limits.

7.13.2: Pie Charts

Control of pie charts is simple. Just select the quantities to be displayed from the list, and a pie is drawn for each of them. To remove a slice from each pie, enter the number of the slice into the text field labeled **Remove Slice**. A value of zero means “remove no slice”.

7.13.3: Scatter Plot

Scatter plots show distributions of data in an x-y graph. By selecting various parameters for the X and Y axes, the correlation of parameters can be assessed graphically.

X Axis

Choose one quantity to be plotted along the X axis.

Y Axis

Choose one or more quantities to be plotted along the Y axis.

7.13.4: Box Plot

Box plots are used to examine the overall structure of the data. Use the list to select the quantities to be plotted, and a box is drawn for each one.

The boxes can be displayed horizontally or vertically, and when they are displayed horizontally, a jitter plot can also be added which shows the distribution of all the data points for each box.

7.13.5: Sunray Plot

Sunray plots show data values distributed around a central point, with the distance of each point from the center being proportional to the data value. This yields a star- or hedgehog-like plot.

Select the quantities to be plotted from the scrolling list; a sunray plot is drawn for each one. The icons along the bottom control how the sunray plots are drawn: with circumference lines, radial lines, and an bounding circle whose radius is the maximum data value.

7.14: Annotation

The **Annotation** popup is used to specify plot parameters that are not dependent on the data, and so do not fall into the category of display settings. These are such things as titles, axis ranges, and so on.

The features of a plot that are independent from the type of plot are called annotation features, and control of these is available through the **Annotation** popup (Figure 7-20). To access this popup, choose **Annotation...** from any plot menu. The **Annotation** popup works over multiple plots in the same way as the display popups. The one difference is that this popup affects all selected plots, regardless of their respective types.

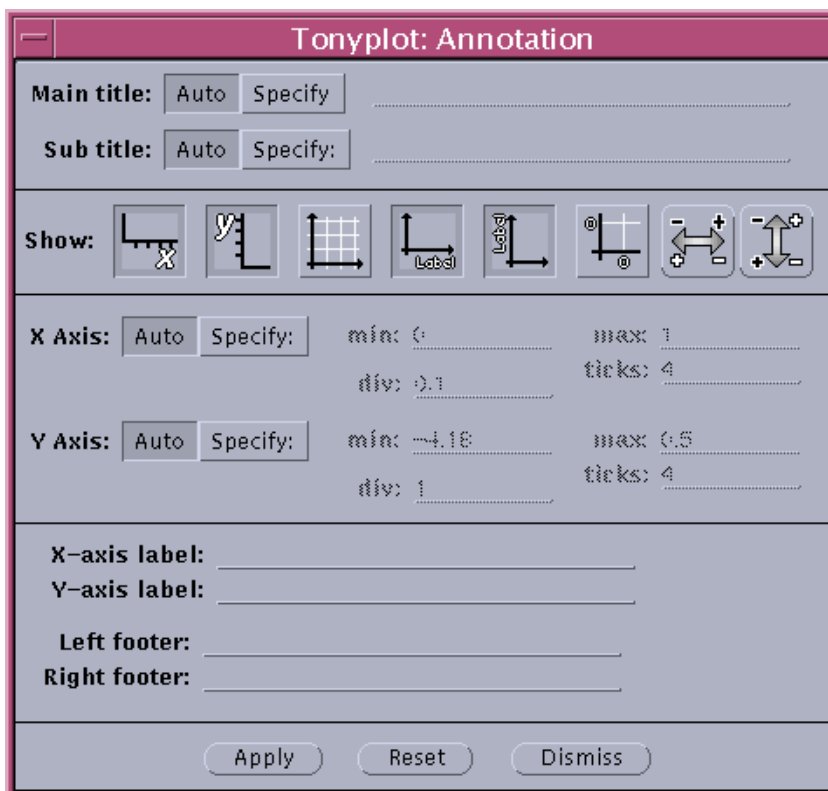


Figure 7-20: Annotation Popup

7.14.1: Titles

Each plot has two titles. TONYPLOT assigns these titles by default, but they can be changed if required. The current main title and current subtitle are displayed in their respective text fields on the popup. The titles can be changed without need for the plot(s) to be redrawn completely by changing the title and pressing the Return key. This leaves all other items as is, and only the titles are redrawn on the selected set of plots.

Note: The choice item must be set to **Specify** in order to change the titles. If **Auto** is selected, titles cannot be changed. This allows more than one plot to be changed with the Annotation popup without the titles on all plots ending up the same.

7.14.2: Show

This item controls features that appear around the edge of the plot. The icons represent, from left to right, x-axis ticks and numbers, y-axis ticks and numbers, grid, x axis label, y axis label, and zero lines. The large buttons can be used to invert the plot, i.e., reverse the positive and negative directions of the x- or y-axis.

7.14.3: Range

The default ranges on the X and Y axes are calculated to ensure that all of the data from all structures in the plot can be seen. These values can be changed however. Selecting **Specify** rather than **Auto** enables the axis control items, allowing the minimum and maximum values to be entered, as well as the division and number of ticks per division. For axes that are plotted on a log scale, the division is always 1.0 regardless of the value entered manually.

When the axis ranges are specifically set, and applied to multiple selected plots, all plots are scaled the same. This allows easy plot comparison of similar data.

7.14.4: Statistics Plots

Some statistics plots do not have the regular range controls as described above. Instead, the controls allow user-specified “bin values” to be entered, which are used when calculating data distributions.

When a statistics plot is selected and the **Annotation** plot summoned, the range controls include a choice, allowing selection of **Auto** bins (automatically determined by TONYPLOT according to the data range), or to specify the bin values. When **Specify** is chosen, use the min and max text fields to enter the minimum and maximum bin values, and then enter the **Number** of bins to be used between these limits. TONYPLOT adds each bin value to the scrolling list when the Return key is pressed on the **Number** line.

7.14.5: Axis Labels

The x axis and y axis labels can be modified. These are only updated on the plot when the **Apply** button is clicked on. Note that once an axis label has been set in this matter, it is always shown, even if the quantity represented on the axis is changed. To return to the normal axis label, erase the user-specified label from this field and click on **Apply** again.

Note: Cross Section plots generated by the Cutline tool will display an automatic x axis label if none is entered manually. This automatic title is provided by TonyPlot according to the type of cutline x axis desired (see the Properties section).

7.14.6: Footers

Any plot can have footer subtitles in the left and/or right corners. No footers are drawn by default, but you can add them with these text fields on this popup.

7.14.7: Special Characters and Macros

Titles, axis labels and footers may all contain “special characters” if needed. These allow alternative letters and symbols to be drawn, such as Greek letters, superscript numbers, etc. The titles and footers may contain “title macros” (see Section 7.19: “Properties”).

7.15: Labels

Labels are used to add arbitrary notes and informative text to any plot. These labels can be drawn with leader arrows or can be free standing. The **Labels** popup (Figure 7-21), accessed by choosing **Labels** from any plot menu, is used to add, change, and delete these labels. As with the **Annotation** popup, the Labels popup is the same for all plot types. It differs in that only the first selected plot is affected, and no others. Each plot has associated with it a list of labels, and each label has a position in the plot to which it belongs. The labels that belong to the selected plot are shown in the list on the **Labels** popup.

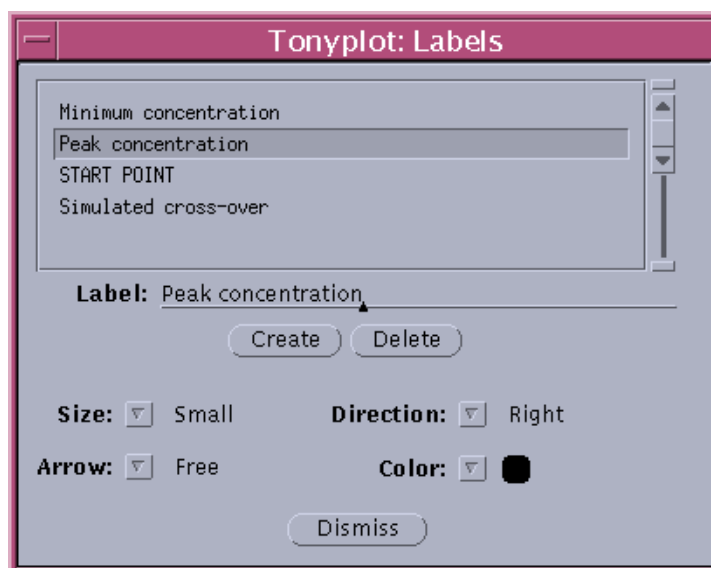


Figure 7-21: Labels Popup

7.15.1: Control Items

Each of the items on the popup are described below.

- **List of labels:** This list contains all the labels that have been defined for the selected plot.
- **Label:** Shows the text of the selected label, and is used to enter new text when creating or changing a label.
- **Create:** Click this to create a new label with the text that is shown in the Label text field. If the field is empty, the new label is created with its text set to **New label**. The label attributes are set from the state of the attribute items (arrow, size, etc). Note that it is possible to have more than one label with identical text.
- **Replace:** This replaces the selected label with new text and/or attributes. Use to change the label attributes such as color, size, etc.
- **Delete:** Clicking on this button deletes the label that is selected in the list. The label is removed from the plot if it has been placed.
- **Arrow:** When placing a label with a leader arrow, the leader can be forced to snap to angles of 45°. This is a “constrained” arrow. A “Free” arrow can be drawn at any angle.
- **Direction:** This determines the direction of the text. The normal choice is “Right” which draws regular text. “Up” and “Down” draw text rotated by 90° upwards or downwards.
- **Size:** This controls the size of the letters in the label. Three sizes are possible: small, medium, and large.
- **Color:** A color palette is provided for selection of the label color. This is used for both the text and the leader line.

7.15.2: Placing Labels

Labels are placed on the selected plot in one of two ways: by clicking to place a simple text-only label, or by dragging, to place a label with leader line (see Figure 7-22).

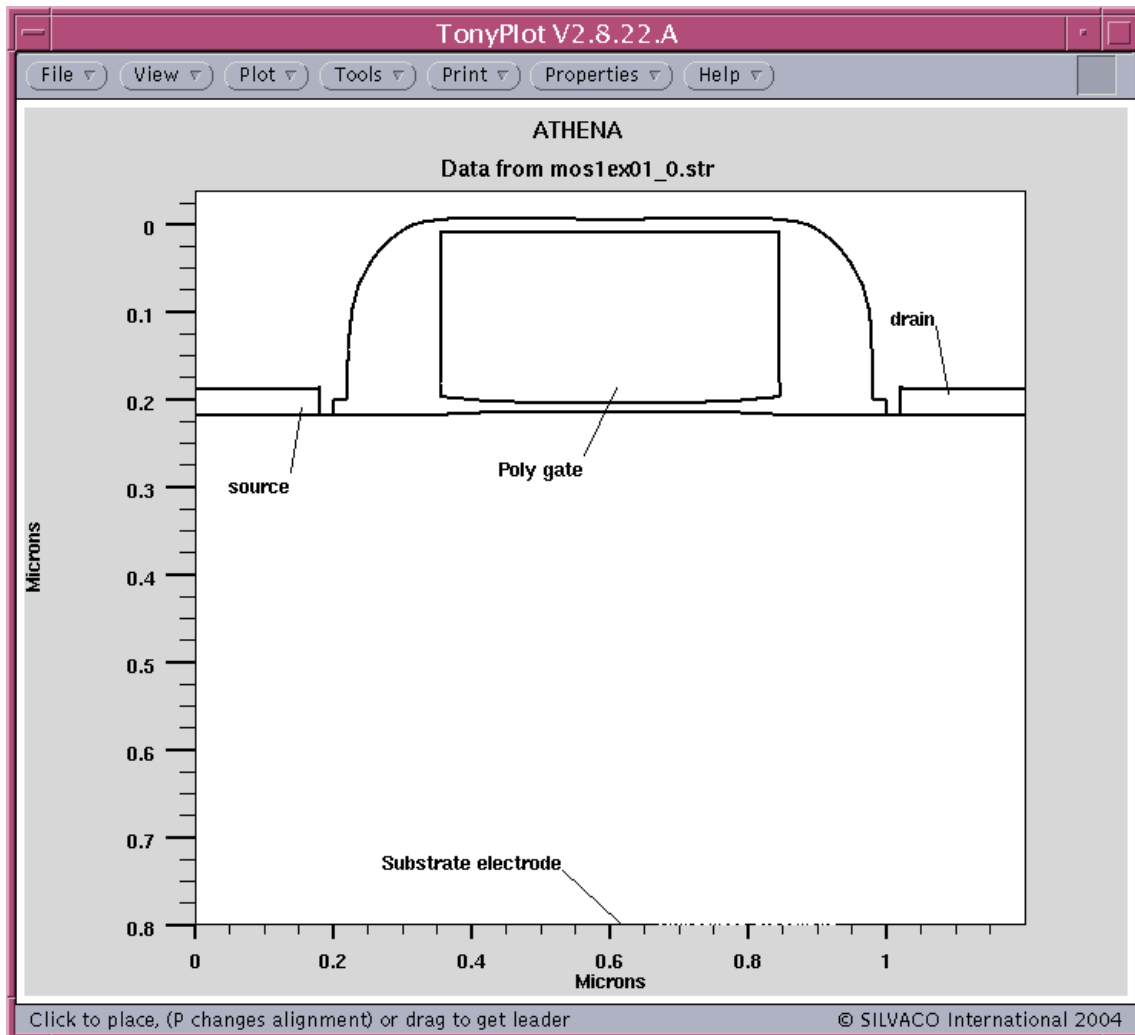


Figure 7-22: Plot With Labels Added

If a leader label is being placed, the start of the drag is where the text appears, and the end of the drag will be the end of leader, i.e., the position to which the leader points. As with any drag operation, holding down the Shift key at any time moves the start point as well as the end point of the drag. When the drag is done, the label text is positioned correctly relative to the direction of the leader line. For example, if the leader points down and to the right, the text is placed so that the leader starts from the bottom right corner of the text.

If a simple text-only label is being placed, the cursor indicates how the text is positioned by pointing to a corner. For example, if the cursor points up and right, the text is placed so that the clicked point is in the bottom left of the text. The cursor can be changed to obtain different alignments by pressing the p key on the keyboard. Four positions are available.

A label can be moved by simply repeating the placing procedure. The old label is drawn in the current background color as a quick erase operation that does not redraw the whole plot. Once placed correctly, a redraw of the view tidys up the display.

Note: If the first character of a label is a space, then TONYPLOT will draw a small “blob” on the end of a leader line. This can be useful in helping to identify the location to which the label refers.

7.15.3: Special Labels

In some cases, TONYPLOT generates labels automatically. If text appears on a plot, it is usually label, placed by TONYPLOT, that can be controlled with the regular label popup as explained above.

Some examples of special labels are:

- **Integration Tool:** This tool (see the Integrate section) adds a label to show the integrated x-range and area. Although placed in a default position, the label attributes can then be customized with the labels popup.
- **2D RSM Plots:** Pressing the ‘v’ key in a 2D RSM plot adds a spot height label to the plot. The label can be moved, but the height does not change, so the label value would then be invalid.
- **Electrode Names:** 2D Mesh structures from ATHENA or ATLAS can contain electrode information. When electrode names are plotted, they appear as labels. By default, they are positioned over the appropriate electrode, but can be moved if desired.

7.16: Tools

As well as displaying information contained in structures, TONYPLOT also has the ability to interrogate that information in a variety of ways. Each method of examining the data is called a **Tool**, and the main **Tools** menu shows all the tools available. The **Tools** menu may show some items as being unavailable. This is because the tool cannot be applied to the current set of selected plots. For example, the HP4145 Emulator tool only works with **Graph** plots, so it is unavailable if no **Graph** plot exists and is selected. Each of the tools are explained in full below.

7.16.1: Cutline

The **Cutline** tool is used on 2D Mesh plots only. It is used to create 1D cross section plots from arbitrary positions within a 2D structure.

Control Items

The **Cutline Tool** popup (Figure 7-23) consists of the following items:

- **Create** — the top section provides different choices for creating cutlines. These choices are free, vertical, horizontal, chained, interface, and keyboard. Each of these is described below.
- **Select** — The center portion allows any cutline created to be chosen for manipulation. This includes shifting and movie making of the cutline.
- **Movie** — The movie section (displayed when the Make movie button is clicked on) is used to make a movie from a cutline by repeatedly moving its position.
- **Shift** — A cutline can be moved once created by using these controls, which are displayed when the Shift position button is clicked on.

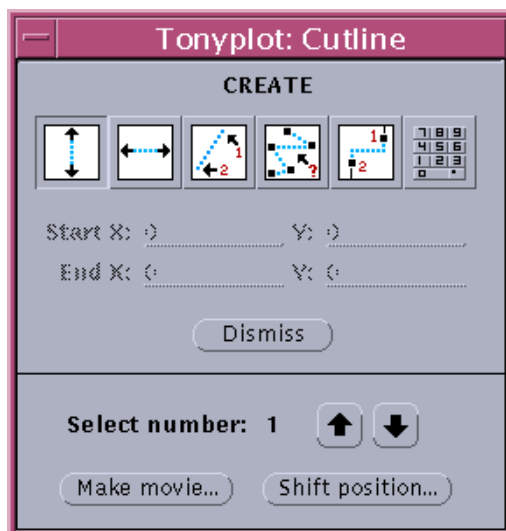


Figure 7-23: Cutline Tool

Creating

To create a cutline, select one of the **Create** options and follow the actions specific to the mode to define the cross-section.

Note: While dragging, holding down the SHIFT key causes the start point to move as well as the end point.

- **Free** — To define a free cutline, drag the mouse over the source plot to define a line through the mesh. Releasing the mouse button creates the cross section.
- **Vertical** — Same as free, but dragging is restricted to a vertical direction. Release the mouse button to create the cross section.
- **Horizontal** — Same as free, but dragging is restricted to a horizontal direction. Release the mouse button to create the cross section.
- **Chained** — To create a chained cutline, click on the mouse **SELECT** button in various places to create a polygon of chain line segments. To erase to last point placed, use the **ADJUST** button. Press the Return key to create the cross section.
- **Interface** — To create this type of cutline, click on the mouse **SELECT** button to place two points on any interface (region boundary). Click on **ADJUST** to erase to last point placed, and Return to see the portion of the interface along which the cross section is calculated. If the portion is wrong, press t to try other routes. When ready to create to cross section, press the Return key once more.
- **Keyboard** — To create a new cutline with exact start and end coordinates, enter the start and end points into the text fields supplied, and click on **Return**. This creates the new cross section from the line so defined.

If more than one mesh plot is selected when a cutline is created, a cross section is calculated for each of them, and the appropriate number of new plots will be generated.

Creating From Multiple Plots

If more than one mesh plot is selected when a cutline is created, a cross section is calculated for each of them, and the appropriate number of new plots is generated.

If a cutline is made from an overlayed mesh plot, the cutline plot generated is also an overlay plot, with each level showing the cutline profile from each level in the mesh plot.

The Section

The cross section created displays the profile of the quantity that was contoured on the mesh, or shows a default profile if no contours were drawn. This new plot can be controlled just like any other cross section plot. All the quantities that were present in the mesh are also available in the cross section. Default titles show the mesh data file from which the cut was made, and the start and end positions of the line.

Interface cutlines create an overlay plot from a single mesh plot. Each level in the overlay represents profiles from one of the materials present at the interface. For example, an interface cutline taken along an oxide/silicon interface creates an overlay cross section with one level showing profiles in oxide, and the other level showing profiles in silicon.

Any other type of cutline produces an overlay cross section if the source plot was an overlay plot. In this case, the new cross section contains one level for each level in the mesh plot. For example, if two meshes alpha and beta are overlaid and a cutline taken, the new cross section plot contains two levels: the first level containing profiles from alpha and the second level showing profiles from beta.

Deleting

To delete a cutline, the cross section plot should be deleted. This removes the cutline from the mesh from which it was created if still present in the view.

Shifting

To shift any created cutline (except interface cutlines), click on the **Shift Position** button. By using the directional arrows on the **Cutline** tool popup, the cutline position is moved up, down, left or right. The amount moved is shown in the **Delta** text fields, which can be modified. To use the shift feature, the mesh plot that contains the cutline must be selected.

Movies

You can create a cutline movie automatically from the **Cutline Tool** popup. It is created simply by moving the cutline position many times and sequencing the resulting cross sections. To create a movie, define the step size and number of steps, and define whether to move the cross section horizontally or vertically. Note that this does not move the actual position of the cutline on the plot as shifting does. To use the movie feature, select the mesh plot that contains the cutline.

7.16.2: Ruler

The **Ruler** tool can be used on any type of plot. It provides coordinate geometry information of any line drawn over a structure. To use the **Ruler** (Figure 7-24), select the plots in which measurements are to be taken, and choose **Ruler...** from the **Tools** menu.

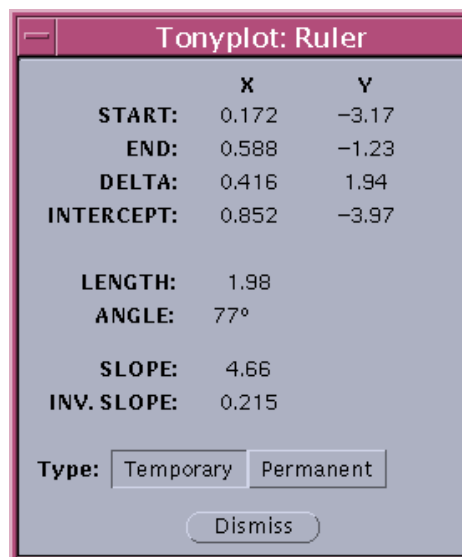


Figure 7-24: Rulers Popup

The ruler position is defined by dragging the pointer across the plot to define a box and line. Holding the SHIFT key down while the pointer is dragged causes the start point of the ruler to move as well as the end point. The Ruler popup shows the following information:

- **Start** — The coordinate of the start point of the ruler.
- **End** — The coordinate of the end point of the ruler.
- **Delta** — The vertical and horizontal distances between the start and end points.
- **Intercept** — The intercept point on the X and Y axes. Terms "X axis" and "Y axis" refer to the lines "y=0" and "x=0" respectively, and not the axes along plot edges.
- **Length** — The distance from the start point to the end point.
- **Angle** — The angle of the end point taken from the start point. Zero is towards positive X, -90 is negative Y, +90 towards positive Y, and ± 180 is towards negative X.
- **Slope** — The gradient of the ruler line.

- **Inv.Slope** — The inverse gradient ($1 / \text{slope}$).
- **Type** — Two types of ruler are available. The default ruler is called a “temporary” ruler, because once the mouse button is released, the lines are removed from the plot (but the values remain displayed in the **Tool** popup). A “permanent” ruler, however, remains in the plot: the temporary ruler lines are drawn in the plot window, and some of the values from the popup are also added at relevant places. This permanent ruler remains on the plot until the ruler is placed once more. Switching the ruler type back to “temporary” also removes a permanent ruler.

To return to normal use of the plot window, the **Ruler** popup must be dismissed.

7.16.3: Probe

The **Probe** tool can be used to look at structure information in a 2D Mesh. This can be useful for debugging simulator output as well as for general use. To use the **Probe**, select one or more 2D Mesh plots and choose Probe... from the main Tools menu. Click anywhere within a structure to activate the probe. A crosshair marker indicated the last position clicked. Measurements are then displayed in the **Probe** popup shown in Figure 7-25.

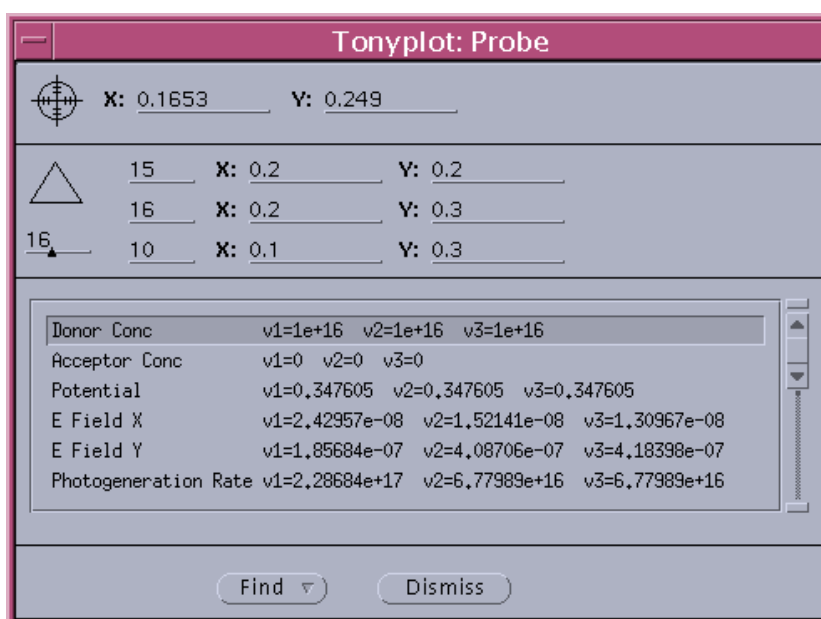


Figure 7-25: Probe Popup

Probe Coordinates — The panel at the top of the tool popup shows the position where the probe was last placed.

Geometry Info — The second panel shows information about the triangle in which the probe was positioned. The internal index is given for the triangle itself, and for each of its three vertices. The actual coordinates of the corners are also displayed.

Impurity Values — The list shows the impurities that are present in the data. Before the probe has been placed, no values are shown, but as soon as the probe is positioned, the values of each impurity is shown at each triangle vertex. The values shown are the actual values (linear scale). There is a property in TONYPLOT that causes the probe to display log values of impurities that are sometimes seen on log scales. See the Properties section for more details.

Find — The **Find** menu allows the probe to work in reverse. Enter the number of the triangle or point to be probed into the appropriate text field and choose the required option from this menu. The triangle is indicated by a brief sequences of flashes, and points are marked by the probe marker

moving to the point on the plot. Alternatively, choosing **Obtuse Triangles** highlights all mesh triangles that contain an angle greater than 90°.

Note: When used with RSM plots (drawn in the 2D mode only), only the probe coordinates and impurity (i.e., RSM output) value are displayed. There is no mesh information available, and the “find” features are not applicable.

7.16.4: Movie

The **Movie** tool in TONYPLOT can allow a group of plots to be combined into an animated sequence which can be viewed in playback like a slideshow. To create a movie, the slides must first be set up. This is achieved by simply creating a group of plots in the main TONYPLOT view, selecting this group, and choosing **Movie...** from the main **Tools** menu. This item is only available if two or more plots are currently selected. A delay is noticed while TONYPLOT creates the movie sequence, and messages appear in the frame footer indicated progress. When complete, the Movie popup (Figure 7-26) appears, showing the first frame of the movie and a group of control items.

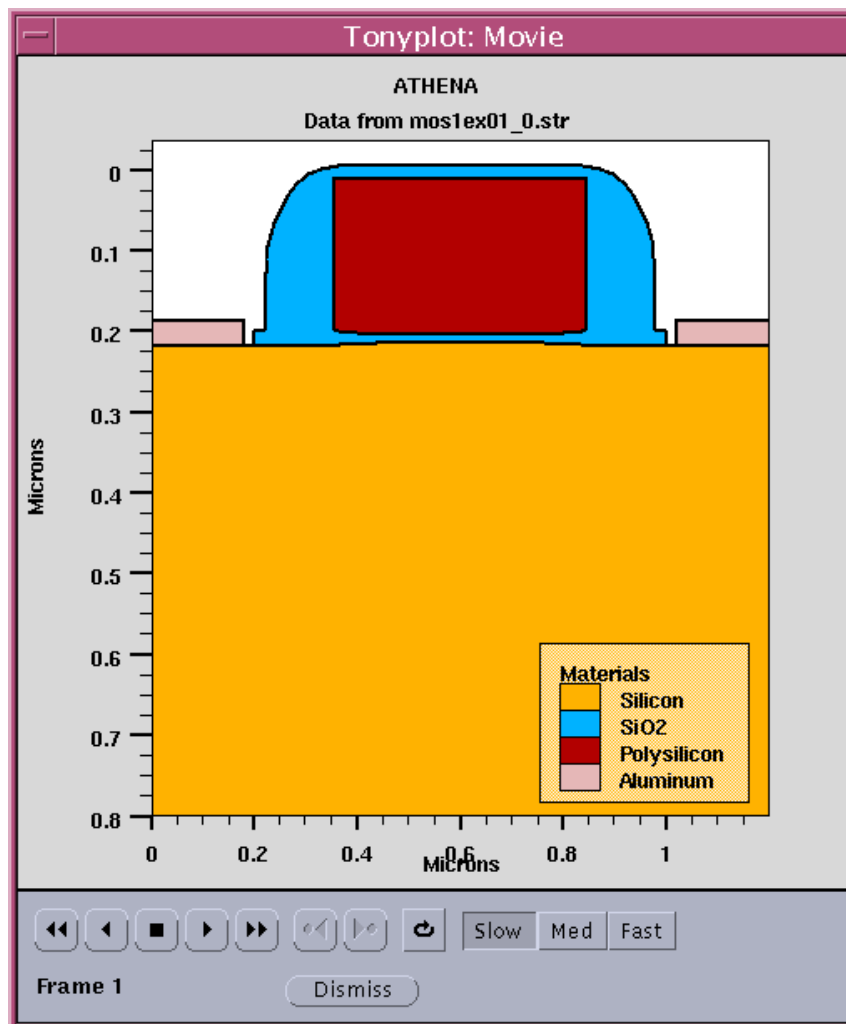


Figure 7-26: Movie Popup

The size of Movie popup can be changed, since this is a TONYPLOT property. This is explained in the Properties section. The control items are:

- **Video controls** — The five buttons perform the following functions: rewind to first frame, play backwards, stop at current frame, play forwards, and skip to last frame.
- **Repeat** — This is the item marked with a looping arrow. When depressed, playback repeats in an endless cycle in the direction determined by the play button pressed.
- **Speed** — Three playback speeds are available. Note that the new speed is not observed until a play button is pressed after the new speed has been selected.

To remove the Movie popup, click on the **Dismiss** button.

Note: Only one Movie tool can be displayed at once.

TONYPLOT can create automatic movie sequences from cutlines without repetitive use of the **Cutline** and **Movie** tools. See the section on the **Cutline** tool for a description of this feature.

7.16.5: HP 4145 Emulator

The HP4145 Emulator is available for any graph plot. Only one plot may be used with the emulator at any one time. When this options is chosen from the main **Tools** menu, the first selected graph plot changes to mimic the output of the HP4145. A HP4145 popup (Figure 7-27) appears containing the controls that emulate the functions of the HP4145.

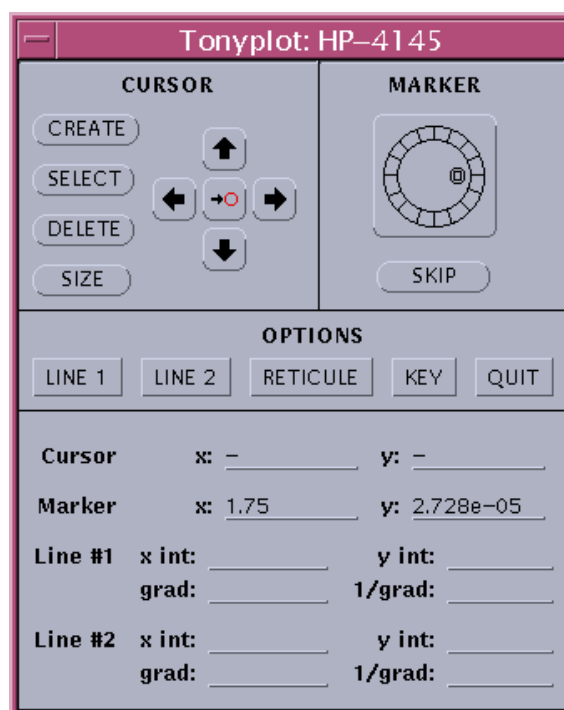


Figure 7-27: HP4145 Popup

The controls of the HP4145 popup are:

- **Cursors** — The control section allows manipulation of cursors represented as plus-sign shaped crosshairs. The buttons allow new cursors to be created, existing ones to be deleted, selection between existing cursors (the current cursor is shown in bold), and the size of the current cursor to toggle between small and full screen. The four directional buttons move the current cursor, and the central button moves the current cursor directly to the marker
- **Marker** — The marker can be moved along its current graph line by moving the dial. To dial is moved counterclockwise by clicking in the left half of the button, and clockwise by clicking in the right half. **Skip** moves the marker from one curve to the next, cycling back to the first curve when the last one has been reached.
- **Options** — Several options can be accessed from the middle panel. **Line 1** and **Line 2** turn on or off a line that joins the marker and cursor. Various geometry information about the lines is displayed on the popup, and on the key. **Reticule** turns on or off the plot grid (the same grid that is shown using the **Annotation** popup), and Key toggles the HP4145 key on and off. **Quit** closes the HP4145 emulator and restore the plot to normal.
- **Information** — The lower panel gives position information for the marker and current cursor, and geometry information for both lines.

7.16.6: Integrate

The **Integrate** tool (Figure 7-28) provides a facility for measuring the area under a single plot curve, or the area between two curves. The X interval over which the area is calculated can be set by positioning marker lines at certain locations along the X axis. The **Integration** tool works with both **XY Graph** and **Cross Section** plots.

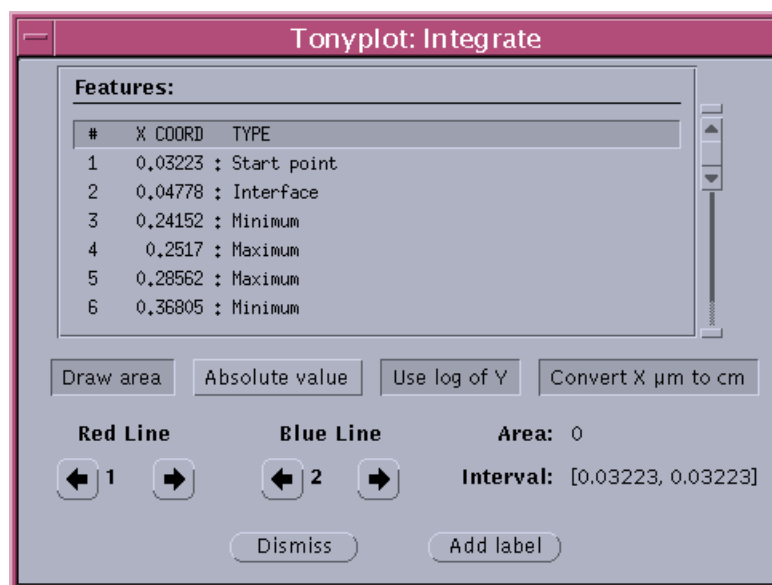


Figure 7-28: Integrate Popup

Features Box

This list shows all the points of interest on the curve in the plot used. It shows all minima and maxima, as well as the start and end X values and positions of all material interfaces. The marker lines, which are used to define the X interval for the integration, can be moved to any of these features, using the arrow buttons, or by using the mouse (explained later). It is possible to use the mouse pointer to add more features; this is explained later.

Options

There are some options that can be toggled on and off when using the **Integrate** tool. These appear in a line under the scrolling list, and each of them is as follows:

- **Draw area** fills the area under/between the curve(s) with a hatched pattern, when turned on. If turned off, no area is drawn, but it is still calculated.
- **Absolute value** uses positive areas only, taking the absolute value of all Y axis values. If turned off, areas below the Y=0 line have a negative area.
- **Use log of Y** calculates the area using log values of Y, rather than the true linear value. This option is independent of the method used to draw the Y axis. In other words, it is possible to draw a curve on a log Y-axis scale, but calculate the area on a linear scale.
- **Convert X um to cm** converts from microns to centimeters. X axis quantities are sometimes plotted in microns (i.e., cross section plots), but Y axis quantities are often given in terms of cm or cm³. Use this option to calculate the area with the X-axis values converted from microns to cm.
- **Results** continually displays the current area and interval on the right.
- **Line control** positions the two lines that specify the interval used for area calculation. These lines can be moved with the buttons marked with left and right arrows. The lines can be placed at any of the features that are shown in the list. A line can be moved directly to any feature by choosing either 'Move RED line to selected' or 'Move BLUE line to selected' from the feature list menu. This causes the appropriate line to move to the feature currently selected in the list.
- **Add Label** creates a label in the plot with the integral information. Depressing the button multiple times updates the label with the latest **Real** and **Interval** values.

Using the Mouse and Pointer

The mouse pointer can be used to move either of the marker lines, and to add new features at any point along the X-axis. To move a line, click the SELECT mouse button anywhere near one of the lines, and then drag the mouse: the line moves to the feature nearest to pointers position. This method allows you to "pick up" a line, move it to a new position, and put it back down.

It is also possible to create new features. To do this, hold down the SHIFT key, and repeat the procedure above. This time, the marker line can be moved to any x location. When you release the mouse button, a new feature is added at the current line position, and the line is moved to it.

7.16.7: Tracers

Tracers are used to illustrate the path of vector fields within 2D Mesh structures. They are drawn as small markers which can be positioned anywhere inside a vector field, and are then animated by TonyPlot to show field strength and direction. The Tracers popup (Figure 7-29) is used to control the positioning and animation of the markers, and choose some options associated with them.

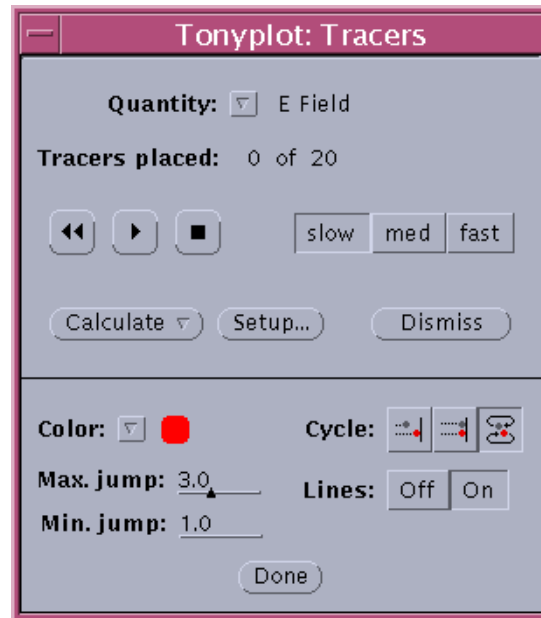






Figure 7-29: Tracers Popup

- **Quantity** — This selects the vector quantity that the tracers should follow. These are the same quantities that are shown on the Vectors popup (accessed from the plot Display popup), but the vectors do not have to be displayed for the tracers to work.
- **Tracers placed** — This indicates the number of tracers that have been placed on the plot, and the number of tracers available.
- **Animation control** — Three video-like controls are provided to control the animation of the markers. These are , which return all markers to their starting points, , which starts the markers, and , which stops them at their current positions.
- **Speed** — There are three speeds available for the animation — slow, medium, and fast. Select the appropriate speed. To change the speed while the markers are moving, press  again.
- **Calculate** — When you have placed all the markers you wish to animate, click on the **Calculate** button. TONYPLOT then traces out the path of each marker. Progress is reported in the lower left corner of the main TONYPLOT frame. When all the paths have been calculated, the markers can be animated. Markers can be placed anywhere within a vector field by clicking at the position where you want a marker to start. The counter on the popup indicates how many have been placed. To remove a marker, press the SHIFT button and click near to the marker to removed: the one nearest the pointer is erased, and if its path has been drawn that too is erased.
- **Setup...** — Press this to see the small panel of setup options available in the **Tracers** tool. These are explained below.

Setup

Click on the **Setup...** button to reveal the options panel on the **Tracer** tool popup

- **Color** — All markers placed use the currently selected color. Different markers can have different colors, by changing the color for each marker placed. Tracer path lines are drawn in the same color as the marker that follows it.
- **Max. jump** — This value controls the “granularity” of the path calculation. Higher numbers reduce the calculation time, but give only approximate paths with long jumps. Smaller numbers produce more accurate paths, but take longer to calculate.
- **Lines** — When turned on, lines are drawn along the tracer paths as the paths are calculated. If turned off, the path is not shown but the tracers still follows the same route.
- **Cycle** — Three cycle modes are available which control the action of markers when they reach the ends of their paths. The first choice stops all tracers as soon as one tracer reached the end. The second choice stops each marker as it reaches the end of its own path. The third option makes each tracer move in a loop, returning to its start point each time it reaches the end.

7.16.8: Poisson Solver

The Poisson Solver (see Figure 7-30) performs an electrical simulation with the 1D structure, and calculate profiles for a set of electrical quantities.

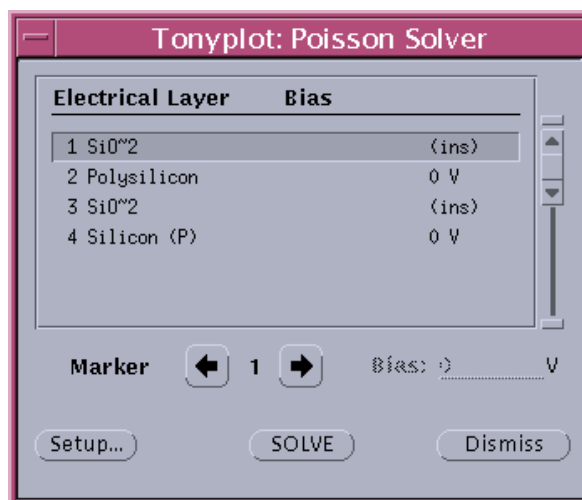


Figure 7-30: Poisson Solver Popup

When the solution is complete, a plot of potential is displayed. Other solutions can be plotted by using the **Plot display** popup. The list of quantities solved is:

- Electron QFL
- Hole QFL
- Electron density
- Hole density
- Intrinsic conc. (nio)
- Potential
- Electron (e-) Mobility
- Hole (h+) Mobility
- Electric Field
- Electrical Conductivity

Some options to control the solver can be changed by clicking on the **Setup...** button, and these options are explained below.

The **Poisson Solver** tool provides a built-in 1D electrical solver, which can be used to perform basic simulations of 1D structures. When it is used, the display of the first selected 1D plot shows all electrical “layers” in the structure, and just one profile (usually “net doping”). An arrow is drawn at the first layer.

The following controls are available:

Layers and biases — The **Poisson Solver** popup shows a scrolling list of all electrical layers within a structure. The layers are areas of the same material; silicon areas are divided up into n-type and p-type silicon. Along with each layer is shown a bias (in Volts) which is applied to that layer when the solver is used.

Marker control — The left and right buttons can be used to move the marker arrow from one layer to the next. The marker arrow is used to select layers for applying an external bias (see below).

Setup... — Click on this button to reveal the **Poisson Solver** options panel (see below).

Solve — Click on this button to perform the simulation with the current options and biases.

Applying A Bias To A Layer

To apply an external bias to any layer, the marker must be moved to any layer that is not an insulator. This can be done either with the left and right buttons on the popup, or by using the mouse pointer to “drag” the arrow into a layer.

The current layer is selected in the scrolling list on the popup. The bias can then be specified by typing the value into the field marked **Bias** on the popup. Press Return to update the list.

For p-type silicon, the bias is converted to a negative value automatically, and to a positive value for n-type.

When all the biases desired have been set, the solver can be initiated by clicking on the **Solve** button.

Setup

The **Poisson Solver**’s setup panel is accessed by clicking on the **Setup...** button on the **Poisson Solver** popup. The following controls are available:

Display Solved Quantities — This list shows all the quantities that the solver calculates. However, only the ones selected are displayed when the solution is complete. All other quantities can be accessed later from the **Plot Display** popup. If other quantities are to be displayed automatically, choose them here. More than one can be selected. Note that this does not affect which quantities are calculated, only the ones that are displayed by default.

Temperature — Specify the temperature to be used for the simulation, or use the automatic default.

FE Mobility — Use this option to activate the field effect mobility option for the simulation.

Work function — Enter a specific workfunction with this option, or use the default value.

SOI Device — Use this option to simulate a device with SOI layers (Silicon On Insulator structure).

Load and Save — It is possible to save the current **Poisson** setup options to disk for use with other TONYPLOTS. Click on the **Save** button to store the current settings. Clicking on the **Load** button retrieves a previously saved setup.

7.17: Printing

Printing in TONYPLOT is available from the **Print** menu which appears on the top portion of the main window. The menu contains four items, outlined as follows:

- **Print view** — This prints the view according to the currently chosen print parameters.
- **Options** — This displays the **Print Options** popup, which is used to alter the type of hardcopy.
- **Printers** — Choosing this item causes the printer editor to be displayed. The printer editor is used to add or modify the printers that TONYPLOT knows about.
- **Forms** — Choosing this item causes the form editor to be displayed. The form editor is used to add or modify the page layouts that TONYPLOT knows about.

The idea behind printing in TONYPLOT is that various levels of decision are abstracted from the user to make the use of possible options as simple as possible. The first level of control is the easiest way to print: just click the **SELECT** mouse button on the button marked **Print** on the main frame. This produces a hardcopy according to the current **Print Options**. The second level of control is provided by setting of the print options. The print options control the major choices associated with printing. The third level of control allows you to inform TONYPLOT of the various printers and page layout forms that are required from the print options. Typically, this third stage is performed only once, since the printers available and the type of paper they accept changes very rarely.

Once you set up the printers (with the **Printer Editor**) and define the various forms (with the **Form Editor**), it is normally unnecessary to use these facilities again. Only the **Print Options** popup is needed for day to day printing tasks.

For the first time user, printing should start at the bottom level, and finish at the top. Once this is done, printing can be done from the top level, occasionally using the middle level to change print options.

7.17.1: Print Options

When a print is made of the current view, TONYPLOT uses the settings that are shown on the **Print Options** popup. These settings can be changed to use different printers, page layouts, files, etc. The **Load Defaults** and **Save Defaults** buttons allow the options to be saved so that the same settings can be used between sessions.

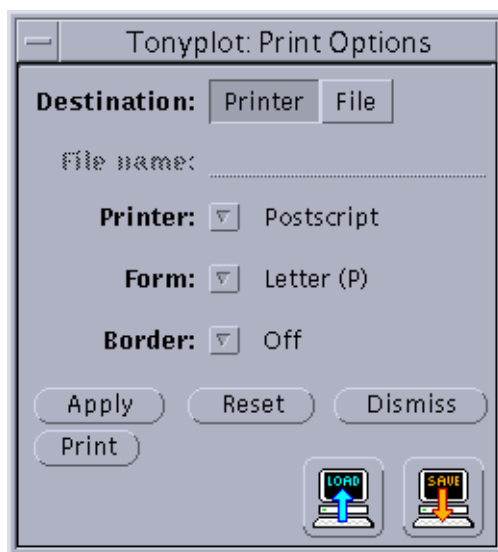


Figure 7-31: Print Options Popup

The items on this popup are as follows:

- **Destination** controls whether TONYPLOT is to produce a print file or send information directly to a printer. If **Printer** is selected, the data is sent straight to the queue to which the printer is attached. If **File** is selected, an intermediate text file is created, whose format depends on the type of printer specified. TONYPLOT selects a default file name if none is supplied.
- **Printer** is used to select a printer from the list of printers that are known to TONYPLOT. By default, only one printer exists. More can be added with the **Printer Editor** (discussed below). The **Printer Editor** is also used to change the printer configurations.
- **Form** is used to select a form from the list of forms that are known to TONYPLOT. By default, only one form exists. More can be added with the **Form Editor** (discussed below). The **Form Editor** is also used to change the form dimensions.
- **Border** is used to turn off the border drawn around each plot. It only affects the printout; borders are still drawn on the screen.
- **Print** provides a printing shortcut. Rather than clicking on **Apply**, and then selecting **Print View** from the **Print** menu, click on this button to perform both actions simultaneously.

7.17.2: Printer Editor

The **Printer Editor** is accessed by choosing **Printers...** from the main **Print** menu. The popup (Figure 7-32) shows a list of all the printers known to TONYPLOT. Also displayed are the configuration details about the printer that is selected from the list.

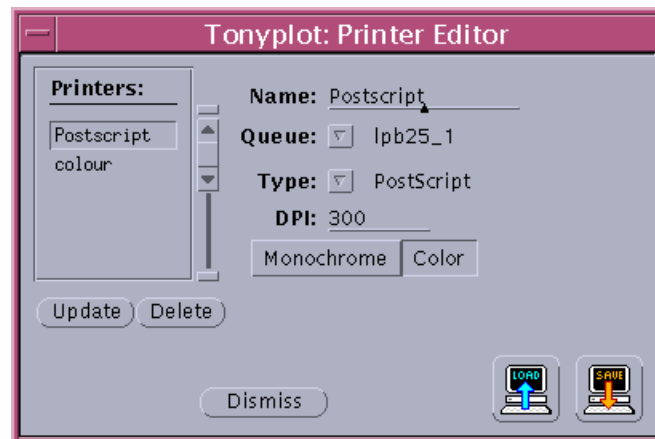


Figure 7-32: Printer Editor Popup

Printers Box

You may use the **Printers:** box to add, change, or delete printers.

- **Adding a printer** — To add a new printer, set printer details on the right and change **Name** to any name that does not already exist. When the **Update** button is clicked on, the printer is added to the list.
- **Changing a printer** — To modify an existing printer, select the printer name from the list, change desired items (except the name) and click on **Update**. Printer settings override existing settings.
- **Deleting a printer** — To delete a printer from the list, select the name of the printer to be deleted and click on the **Delete** button. The name and printer are removed.

As printers are added and deleted, the **Printer** item on the **Print Options** popup is updated so that it always reflects the latest list of known printers. TONYPLOT reads system print information from the standard lpstat service available on all platforms. You can set the environment variable `PRINTER` to add a specific printer to the list or when the lpstat service is unavailable.

Controls

The following controls are available:

Name — The name of the printer. This can be any string, but each printer must have a unique name.

Queue — This item shows all the printer queues that TONYPLOT found in the printer file. Select the queue to which the printer is attached. If the queue is not present, it is necessary to print to files rather than direct to the printer. The files can be sent to the printer manually at a later time.

Type — This shows the type of information that the printer understands. Select the format desired. Note that some formats are suitable only for printing to a file, as no printer supports those formats. For example, PCX format is an image file format and not really a printer input format.

DPI — Indicated the dots-per-inch resolution of the printer. This is not used for all printers.

Color — Selects whether the printer produces black and white (monochrome) images only or whether it supports color.

The **Load Defaults** and **Save Defaults** buttons exist so that the printers in the printer editor can be retrieved between sessions.

7.17.3: Form Editor

The **Form Editor** performs a similar function to the **Printer Editor**, but stores page layout rather than printer details. All page layouts known to TONYPLOT are shown in the list on the **Form Editor** popup (Figure 7-33).

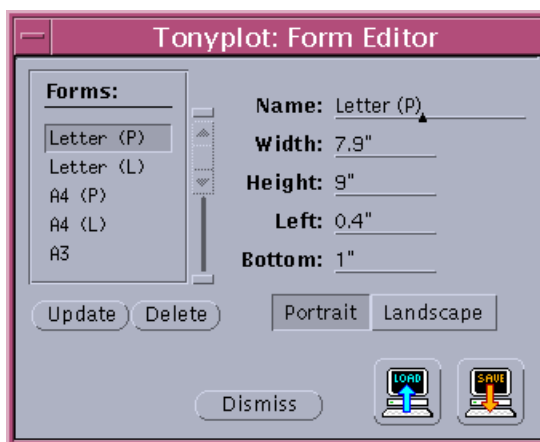


Figure 7-33: Form Editor Popup

Adding, deleting, and modifying forms is done in the same way described for printers (see previous section). The items on the popup are:

- **Name** — The name of a form can be any string, but each form must have a unique name.
- **Width & Height** — Indicates the size of the print image (not the size of the paper). Numbers are interpreted as inches unless proceeded with the letters “cm” to indicate centimeters.
- **Left & Bottom** — Indicates the margins between the left and edges bottom edges of the paper and image. Numbers are read as inches unless proceeded with the letters “cm” to indicate centimeters.
- **Orientation** — The choice item indicates whether the form is upright (portrait) or sideways (landscape). TonyPlot will rotate plots by 90° when printing to landscape forms.

The **Load Defaults** and **Save Defaults** buttons exist so that the forms in the form editor can be retrieved between sessions.

7.17.4: Printing At Startup

TONYPLOT can make hardcopies (or print files) when first invoked by using a combination of the `-print`, `-printer`, and `-form` options. The `-print` option must be given. If it is supplied as the last argument to TONYPLOT, then a default name is used for a print file. If it not the last argument however, the argument that follows it is taken as the name of the file to be created.

If no `-printer` option is used, TONYPLOT assumes a monochrome Postscript printer and 300 DPI. If something other than this is required, the `-printer` argument can be given, with the name of the printer to use (as defined with the **Printer Editor** discussed previously).

If no `-form` option is used, a default Letter size portrait form is used. If another form is required, use the `-form` argument with the name of the desired form, as defined from the **Form Editor**.

7.17.5: Queues and Printers

When TONYPLOT creates a hardcopy, it converts the plot shown on the screen into a text file which can be sent to a printer for printing. Each type of printer accepts different types of files, and so TONYPLOT needs to know the printer being used. Printers read these files from an area known as a “queue”, and TONYPLOT puts the files it creates into this queue. Each different printer reads from a different queue. Therefore, you must tell TONYPLOT which queue the printer is using. Attaching printers to queues is a task performed only upon installation of the printer, and if you have any questions about available printers and queues, or need to find out which printer is attached to which queue, you should contact your system administrator.

7.17.6: System Configuration

The method your system uses to configure the various printer queues is independent of the operating system you are using. TONYPLOT is independent of uses the lpstat service available on all-supported platforms: Linux, Solaris, and HP-UX.

7.17.7: Adding Printers To TonyPlot

To add a printer to TONYPLOT, the **Printer Editor** is provided. Display this now by choosing **Printers...** from the main **Print** menu. TONYPLOT comes with a printer built in. This may or may not be useful at any particular site. We can use this built-in printer as an example.

The list on the left shows the names of the printers TONYPLOT knows about. The one that is selected from this list has its various set up details shown on the right. This printer is attached to a queue called “lp”, if that queue exists. If not, it shows the first queue TONYPLOT finds. Underneath the queue is the type of printer. This is “PostScript” by default, but TONYPLOT can write hardcopy files for many other types of printer, too. Next is “DPI”, which is the number of dots per inch the printer can achieve. Last, a choice item indicates whether the printer is a monochrome printer, or a color printer.

Assume that the printer in question is a DeskJet C500 printer, and that it is attached to a queue called “djet1”. To tell TONYPLOT about this printer, perform the following steps:

1. Enter a name to use for this printer. The name can be anything, but it is best to use something useful. Call the new printer “Deskjet” — enter the name `Deskjet` into the space marked “Name:” (erase the name that is already there).
2. Choose the queue to which the printer is attached. TONYPLOT finds all the queues that can be used, and the one you want should be chosen from the choice item marked “Queue”. For example, select “djet1”. If you are not sure which queue to use, consult your site administrator.
3. Now tell TONYPLOT the type of printer you are adding: this makes sure the hardcopy files are created correctly. Choose “Color DeskJet 550”, because that is what is being added.
4. Set the DPI by finding the value as given in the manual for the printer. Looking it up, find the value to be 600, so enter 600 into the space provided.

5. Because this printer produces color plots, tell TONYPLOT to produce hardcopy files with color information. Set the choice item to “Color”.
6. Now the new printer has been defined. Now add this to TONYPLOT’s list by clicking on the **Update** button. The name “Deskjet” then appears in the list.

This process can be repeated as many times as needed, to enter all the printers that are to be used with TONYPLOT. Once complete, click on the **Save as defaults** button so that information is saved and will not need to be re-entered the next time TONYPLOT is started.

Because the information about the printers is unlikely to change, it is unlikely that the **Printer Editor** is used often.

7.17.8: Adding Forms To TonyPlot

To add a form to TONYPLOT, the **Form Editor** is provided. Display this now by choosing **Forms...** from the main **Print** menu. TONYPLOT comes with several form built in, and these are OK for use with any printer. If not, a new form can be added.

The default forms are:

- **Letter (P)/A4(P)** selects letter-sized paper in portrait orientation (height larger than width).
- **Letter (L)/A4(L)** specifies letter-sized paper in landscape mode (width larger than height)
- **A3** selects paper which is twice the size of Letter paper. Only a portrait version is supplied.

The list on the left shows the names of these forms The one that is selected from this list has its various set up details shown on the right. When adding a new form, the name can be anything, however it is preferable to choose a useful name. The various page dimensions are displayed by TONYPLOT in inches, although they can be given in centimeters.

Let us assume, a new form needs to be set up and printed on unusual sized paper, for example, a 12- by 14-inch page with a 1-inch margin on all four sides. To add this to TONYPLOT:

1. Select and enter a name for the form. It can be anything, preferably something useful. Call the new form “Large” — type in the name “Large” into the space marked **Name:** (delete the name that is already there).
2. The “Width” and “Height” are the area dimensions into which TONYPLOT is allowed to draw. For our example, the width is 10" (12" minus left and right margins of 1" each). Enter 10 into the Width field. The height is 12" (14" minus top and bottom margins of 1" each), enter 12 into the Height field.

Note: Note: if the sizes are in centimeters, add the letters “cm” to the number, e.g., “12 cm”.

3. Specify the margins by entering the Left and Bottom margins; enter “1” for each.
4. Next, TONYPLOT must be told whether the form is a portrait or landscape form. On portrait form, the image appears vertically oriented (height longer than width). On landscape, it appears rotated by 90°, and the image is drawn sideways on the paper with the width longer than the height.
5. After the new form is defined, add this to TONYPLOT’s list by clicking on the **Update** button. The name **Large** then appears in the list.

You can repeat this process as many times as needed to enter all the forms that can be used with TONYPLOT. Once complete, click on the **Save as defaults** button to save this information and so that you do not need to re-entere the next time TONYPLOT is started.

The information about the forms is unlikely to change, and it is not likely that the **Form Editor** is used often.

7.17.9: Setting Print Options

Once the desired printers and forms have been entered and saved, only the **Print Options** popup is needed to change the way TONYPLOT creates hardcopies. The **Print Options** popup is displayed by choosing **Options...** from the main **Print** menu. Do this now to note how the popup looks.

The first choice on the popup is labeled **Destination:**. This instructs TONYPLOT to either create a printout directly or to create a hardcopy file instead. Choose either **Printer** or **File**. When **File** is selected, specify a filename. If a filename is omitted, a default filename is provided.

Regardless of the chosen destination, tell TONYPLOT the name of the printer that is to be used. If the destination is set to **Printer**, the hardcopy appears on that printer immediately; if set to **File**, the file produced is suitable for sending to that printer at any time. The names of printers that can be selected are the same as those seen in the **Printer Editor**. It should be clear that if the printers have been set up correctly with the **Printer Editor**, from now on only a name need be selected from this popup.

As well as choosing a printer, it is necessary to choose a form to which the hardcopy is scaled. These names are the names seen on the **Form Editor**. Again, it should be clear that as long as all the forms are entered correctly with the **Form Editor**, from now on only the appropriate name is needed to be selected.

Once the options are set, the changes must be applied. to do this, click on the **Apply** button. To create a hardcopy now, click on the **Print** button (this also applies the changes; it is not necessary to press both buttons). To create a hardcopy later, choose **Print View** from the main **Print** menu, and the same options are used.

If using the same options again is desired, save the settings by clicking on the **Save as Defaults** button. These options are used whenever TONYPLOT is restarted.

7.18: Printer Drivers

7.18.1: Stack Size

When creating a PostScript file, TONYPLOT assumes the stack size for the PostScript interpreter to be 200. For example, no polygons of greater than 200 points are drawn — lines longer than this are split into smaller sections. If this default value is too large, it can be changed by setting the value of an environment variable `GRF_STACKSIZE` to the maximum for the interpreter being used. This must be done before TONYPLOT is started. It has no affect on a TONYPLOT currently running.

Example

```
% setenv GRF_STACKSIZE 100
% tonyplot bias.log
```

7.19: Properties

There are many aspects of TONYPLOT's behavior that you can alter to suit your needs. These are referred to as TONYPLOT Properties, and they exist to tailor the characteristics of various operations. All properties can be viewed and modified using the **Properties** popup, which is displayed by choosing a category from the **Properties** menu on the main window. Once the popup is visible, any of the other categories can also be looked at by choosing one from the item in the top portion of the controls. Some categories are slightly apart from the main group: these are **Materials** and **Functions**. The **Functions** option displays the **Functions** popup, the same popup accessible from the **Plot Display** windows. The **Materials** popup is described at the end of this chapter. Each category is explained in detail and shown in Figures 7-27 through 7-30.

7.19.1: Drawing Options

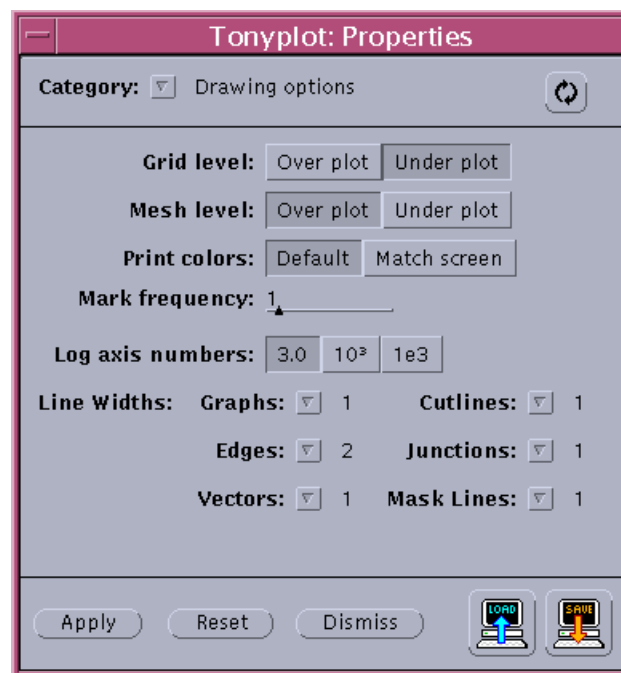


Figure 7-34: Drawing Options

- **Grid level** — This controls whether the axis grid is drawn on top of the displayed structures, or underneath them. The axis grid is controlled from the **Annotation** popup.
- **Mesh level** — This controls whether the simulation mesh is drawn on top of the displayed structures or underneath them. The simulation mesh is controlled from the **Display** popup for either **2D Mesh** plots or **XSection** plots.
- **Print colors** — Control colors used when creating color hardcopies. If **Default** is chosen, the standard print color is used. These are the same as the color used on the screen but on a white background with a black foreground. Choosing **Match Screen** overrides this, and print colors exactly match the screen colors.
- **Mark frequency** — For line plots, controls the number of marks or points that are drawn along the curve. 1 draws a mark at each and every data point, while any other value draws marks at the specified frequency.
- **Log axis numbers** — When log axes are displayed on either **XSection** or **XYGraph** plots, the way numbers are drawn depend on this item. The number 3 is used as an example.

- **Line widths** — This group of controls sets the thickness of the lines used to draw **Graph Lines**, **Edges** (2D Mesh, XSection), Vectors, Fonts, and Cutline positions. 1 represents normal thickness, with 2 to 4 representing increasingly thicker lines.

7.19.2: Plot Options

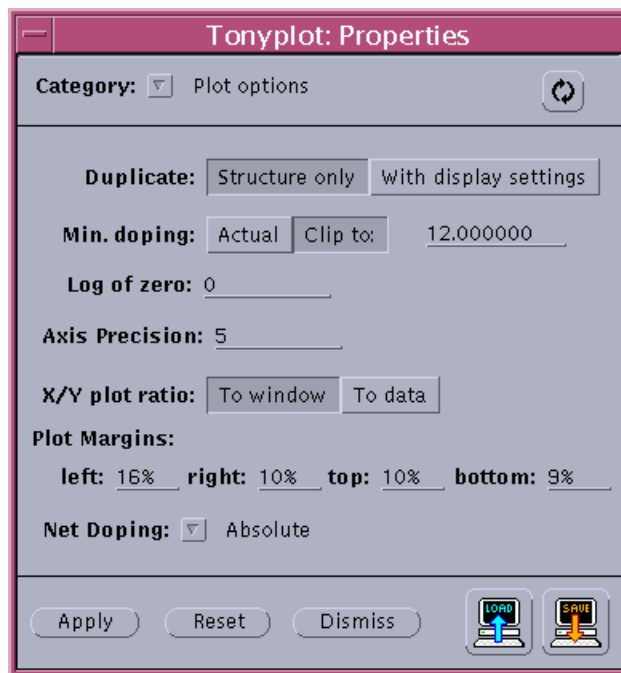


Figure 7-35: Plot Options

- **Duplicate** — When a plot is duplicated, the display settings may optionally be applied to the new plot. This causes the new plot to look identical to the original. If **Structure Only** is chosen, a default display setting is used for the new plot.
- **Minimum doping** — Doping concentrations can either be displayed to their actual minimum values (choose **Actual**) or to a specified value (choose **Clip to**), with the value in the text field indicating the minimum value at which doping is clipped.
- **Axis Precision** — When required, the axis ticks are rounded up to the maximum number of digits (**Axis Precision**) to avoid cluttering the plots.
- **Log. of zero** — When plotting results that involve the logarithm of zero (in any base), TONYPLOT uses this predefined number as the result. Zero is used by default.
- **XY plot ratio** — Controls the relative scaling in the X and Y directions of the data. If **To Window** is chosen the axes scale independently so that the whole window is occupied. If **To Data** is chosen, the axes scale together so that the aspect ratio of the data is maintained. In this mode, at least one of the axes span the subwindow.
- **Plot margins** — Controls the space between the edges of the plot and the edges of the window. These are specified in terms of a percentage of the window dimension (window width for left and right and window height for top and bottom).
- **Net Doping** — Controls whether the absolute or signed values of the net doping are used in plots (absolute or n/p types).

7.19.3: Main Window

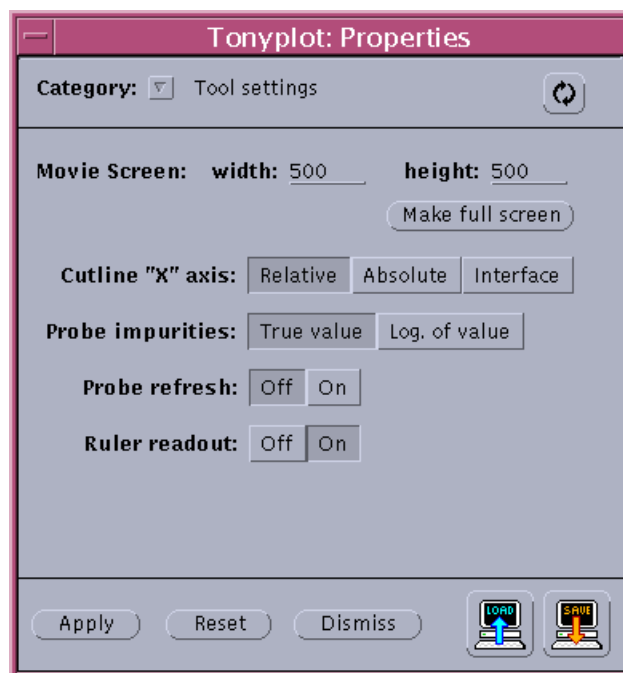


Figure 7-36: Main Window Properties

Layout selects the style used for laying out subwindows within the view. The first two options represent **Multiple** mode (with either horizontal or vertical preference), the third is **Palette** mode (where one plot is larger than all the others), the fourth is **Page** mode (only one plot is shown at a time), and the fifth is **Scattered** (which allows you to move and resize the windows).

When in **Scattered** mode, windows can be resized by dragging their edges. Dragging a corner allows resizing in both the X and Y directions at the same time. To move windows, hold down the SHIFT key, and then drag the edges or corners. With a combination of moving and resizing, any window configuration can be created.

Frame Size sets the size of the main TONYPLOT window. If set to **Custom**, any size can be given. If the properties are saved (with the **Save Defaults** button), then the custom size is used each time TONYPLOT is started.

Panner Jump sets the amount of new plot exposed when a zoomed plot is panned with the zoom panner. The fractions shown are fractions of the window size.

7.19.4: Tool Settings

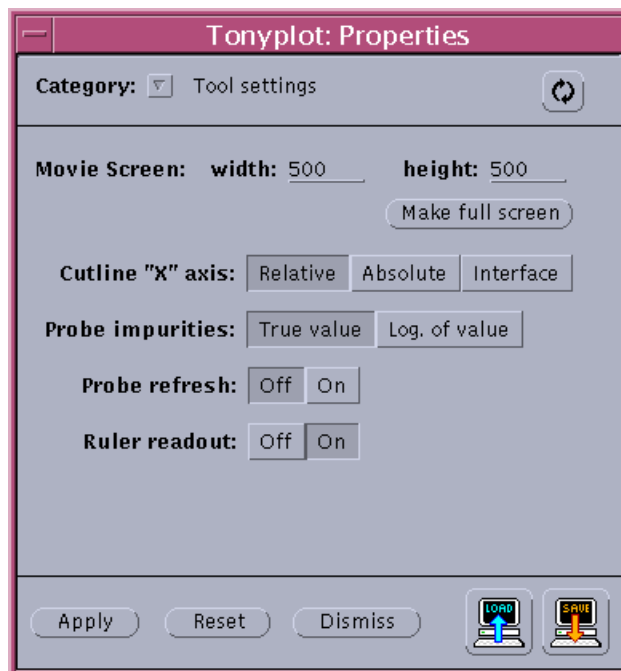


Figure 7-37: Tool Settings

- **Movie screen size** — The dimensions of the **Movie** tool playback window can be customized here. Click on the button marked Make Full Screen to automatically enter sizes that match the size of the entire screen. These sizes do not affect any movie currently displayed, but only movies created after this change has been applied.
- **Cutline X axis** — There are three methods for marking the X axis on cutline profiles. By default, the X coordinate is relative to the start of the line, and so measures distance along the line. If **Absolute** is chosen, the axis shows the X coordinate of the original mesh (if the cutline is horizontal) or the Y coordinate (if the cutline is vertical). Cutlines that are neither horizontal or vertical show a **Relative** X axis. If **Interface** is chosen, the X axis is centered around the first material interface in the cross section, i.e., the first interface is at X=0. If no interface exists, a Relative X axis is drawn.
- **Probe impurities** — This controls how impurity values reported by the **Probe** tool are shown. By default, true linear scale values are shown. Select **Log of value** to see the values on a log scale.
- **Probe refresh** — Switch probe refresh on to remove old probe targets from the plot area.
- **Ruler readout** — When this is turned on, the position of the pointer is displayed in the frame footer whenever the rules is being used.

7.19.5: Overlays

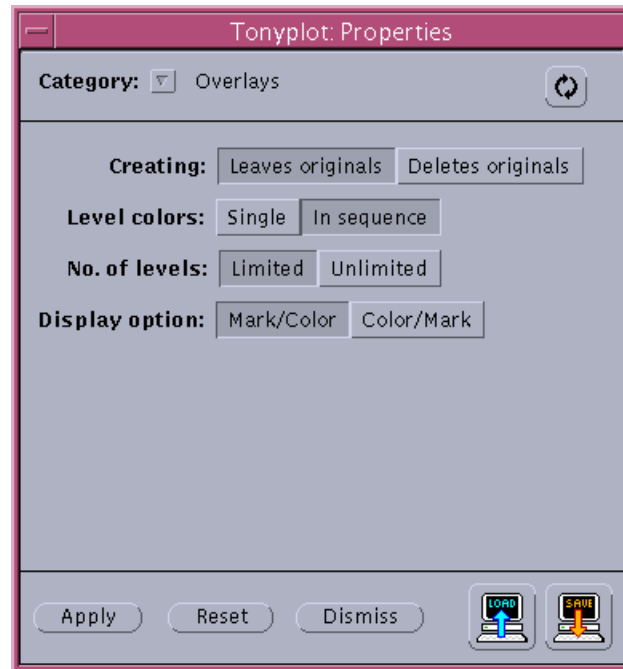


Figure 7-38: Overlays Settings

- **Creating** — When overlays are created, the plots from which it is comprised can either be left in place (choose **Leaves Originals**) or deleted when the overlay has been drawn (choose **Deletes Originals**). If the originals were deleted, they can be retrieved later by splitting the overlay. If they were not deleted, splitting the overlay duplicates the originals.
- **Level colors** — Overlay plots can show each level in the same color (if **Single** is chosen), or use a different color for each (if **In Sequence** is chosen).
- **Number of Levels** — To avoid confusing plots, the number of levels that can be added to an overlay is limited. Choose “unlimited” to deactivate this feature.
- **Display option** — Allows the choice of whether to use mark types for quantities plotted and color for the level, or to use mark types for the level and color for the quantity.

7.19.6: General Colors

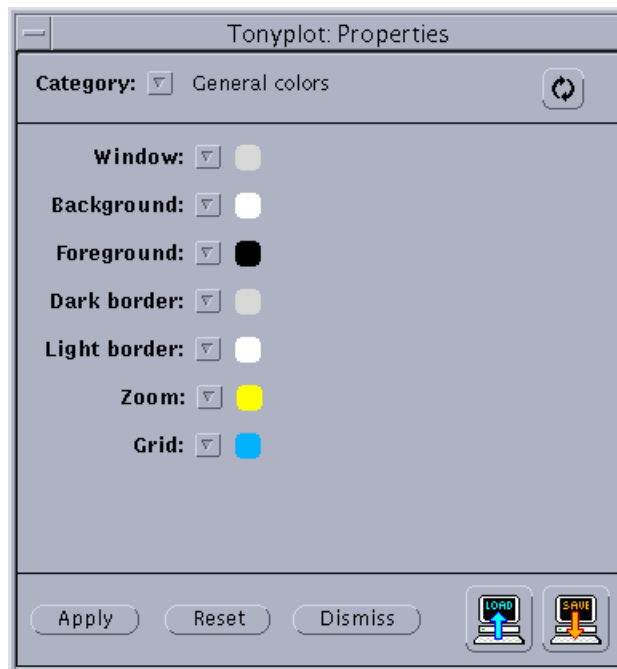
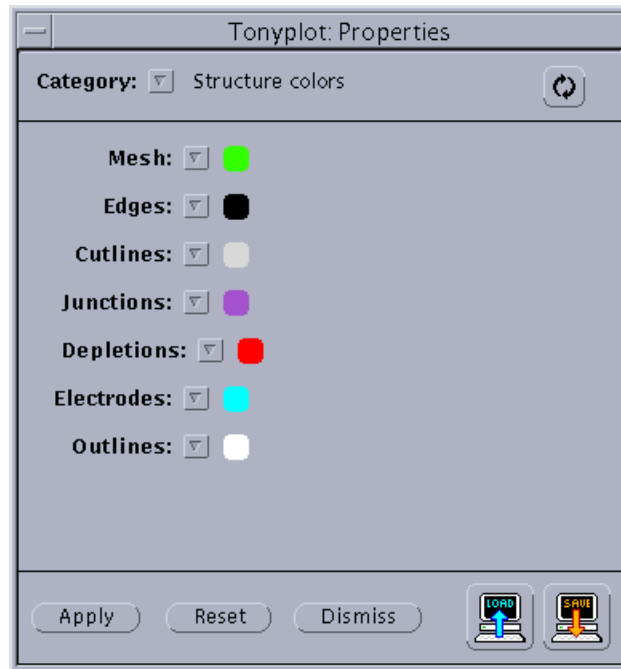


Figure 7-39: General Colors Settings

General colors are colors of items that are not related to any specific data type. Each can be specified independently, and set to any color that TONYPLOT supports.

- **Window** — The color of the subwindow.
- **Background** — The color of the plot background.
- **Foreground** — The color of the plot foreground.
- **Dark border** — The color of borders on unselected plots.
- **Light border** — The color of borders on selected plots.
- **Zoom** — The color used when dynamically defining areas, lines etc. on the plot. Examples are: zooming and placing cutlines. The color specified here may not be the color observed if the color on which the lines appear is not the actual **Background** color. This is due to the graphics operation used. If it is found that the defining lines are hard to distinguish from the color on which they are drawn, changing this color may improve the contrast.
- **Grid** — The color of the axis grid.

7.19.7: Structure Colors



Structure colors are colors of items related to specific data items. Each is specified independently and set to any color that TONYPLOT supports.

- **Mesh** — The color of the simulation mesh on 2D Mesh plots and Cross Section plots.
- **Edges** — The color of edges on 2D Mesh plots and Cross Section plots.
- **Cutlines** — The color of cutline positions on 2D Mesh plots. Note that the color specified here may not be the color observed if the color on which the line appears is not the actual **Background** color. This is due to the graphics operation used. If distinguishing the cutline position line from the color on which it is drawn is difficult, changing this color may improve the contrast.
- **Depletions** — The color of Depletion Edges on 2D Mesh plots.
- **Junctions** — The color of Junction on 2D Mesh plots.
- **Electrodes** — Color of hatching used to indicate electrodes on 2D Mesh plots.
- **Outlines** — The color of contour outlines.

7.19.8: Sequence Colors



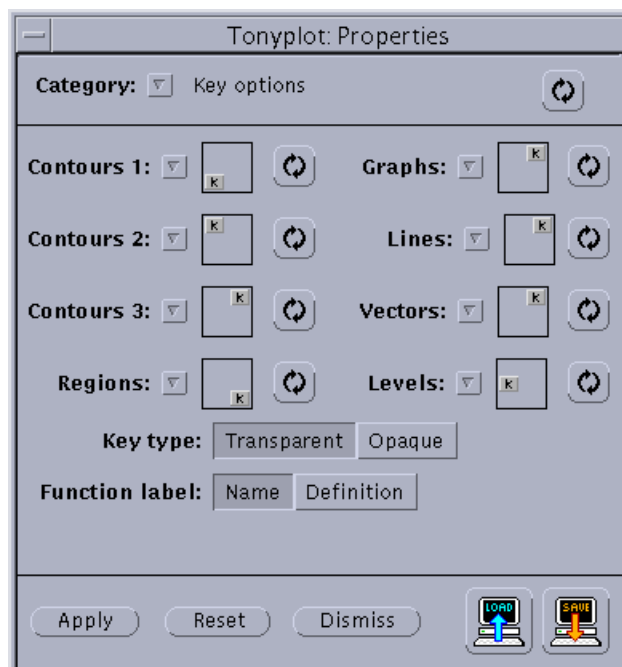
TONYPLOT uses sequence colors whenever a group of items are plotted and each needs its own color. Examples are: lines on Graph and Cross Section plots, regions, levels, and so on. The colors used are defined by the list of sequence colors indicated on this popup. If the first color is red, for example, the level one is red, the first cross section profile is red, region 1 is red, and so on. If more than twelve colors are needed, the color sequence repeats at one.

7.19.9: Sequence Marks



Graph lines are represented by Lines and Marks. TONYPLOT uses different mark types to represent either quantities or levels. These can be defined in this popup. The types are: **Cross**, **Circle**, **Plus**, **Triangle**, **Square**, and **Star**. This popup also allows changing of the size and width of the marks, from a scale of 1 to 6 in size, and 1 to 3 in width.

7.19.10: Key Options



Each key position item provides eight options. If you select **Off**, then that key will not be drawn. If any of the six specific positions is selected, the key is drawn in that position in the plot. If the icon with the arrow is displayed, then the key has been positioned by hand and is in none of the six standard locations.

- **Contours** — There are three contour keys: one for each set. There are three items to control each one separately.
- **Regions** — Used in 1D and 2D Mesh plots to indicate the color used to distinguish each material region or region parameter.
- **Graphs** — The Line key for any graph plot, showing line colors and mark types.
- **Stats** — Key for identifying information on Statistics plots.
- **Vectors** — Vector key for 2D Mesh plots.
- **Levels** — Level key for overlay plots.
- **Key type** — This sets the ways the keys or legends are drawn in plots. **Transparent** (the default) allows the key box to show the plot underneath. **Opaque** covers over any part of the plot under the key.
- **Function label** — Determines whether functions appear on key legends as names (i.e., Function 1) or as their definitions, as shown on the Functions popup.

7.19.11: Environment

The screenshot shows the 'Tonyplot: Properties' dialog box with the 'Environment' category selected. The fields are as follows:

Field	Value
Name of user	Frederic Loranger
Login name	loranger
Group	silvaco
Host	mrquiet
Company name	
Project	

At the bottom, there are buttons for 'Apply', 'Reset', and 'Dismiss', along with two icons representing different user profiles or environments.

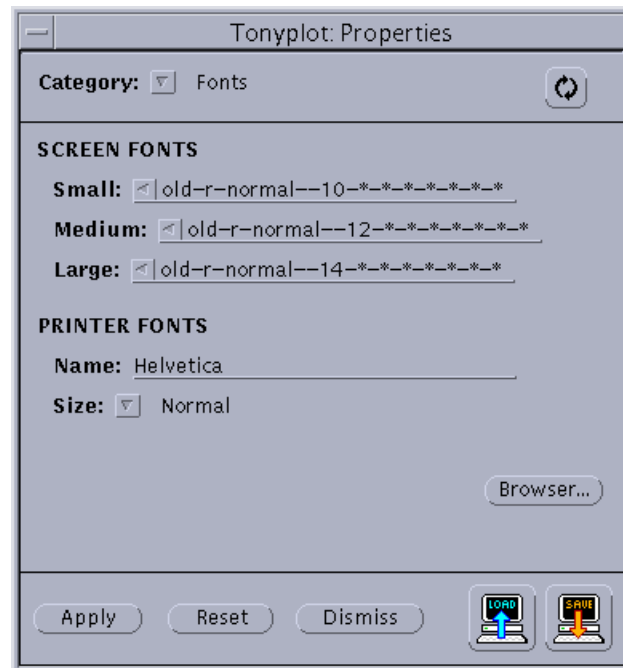
- **Name of user** — This is your name as known by the operation system when TONYPLOT starts. This can be used in titles and footers by using the macro \$NAM.
- **Login name** — This is the login name for the account currently being used. This can be used in titles and footers by using the macro \$USR.
- **Group** — This is the login group name of the account currently being used. This can be used in titles and footers by using the macro \$GRP.
- **Host** — Name of the workstation being used to run TONYPLOT (which may not be the workstation used to display TONYPLOT). This can be used in titles and footers by using the macro \$HST.

- **Company name** — This is the name of your company (if any). This information is not known by TONYPLOT, so is blank by default. This can be used in titles and footers by using the macro \$COM.
- **Project** — This is the name of the project you are currently working on. This information is not initially known by TONYPLOT, so is blank by default. This can be used in titles and footers by using the macro \$PRJ.

As mentioned above, some of the environment properties have an associated macro. For example, your name can be represented with \$NAM. This macros can be used in titles, labels and footers. Other macros are also available: \$DAT shows the current date, \$TIM the current time, and \$PWD the current working directory.

7.19.12: Fonts

The fonts used by TONYPLOT on the screen and on the printer (or in print files) can be selected to any font available on the corresponding output device. There are text fields to allow entry of the desired font names, and a browser which allows any font to be picked from a list of available fonts.



Screen fonts — TONYPLOT can not scale fonts used on the screen, so three fixed sizes are used, and a font must be specified for each.

Printer fonts — For the printer (or printer file), only one font is needed since printer fonts can be scaled to the exact size needed. However, for some presentation formats the font may be too small to read clearly. To enlarge the font relative to the plot size, select a size from **Normal**, **2 column** or **1 column** (for displaying a plot in a single column or across two columns respectively).

Browser — Click on the **Browser...** button to display a font browser. This shows a list of all available screen fonts. To choose any font, select the line from the list and choose the desired **Apply** option. The filter can be used to reduce the number of fonts displayed in the list. TONYPLOT cannot determine the printer fonts that are available so the browser cannot be used to select one: you must do this yourself and enter a valid font name into text field on the **Properties** popup directly.

7.19.13: Miscellaneous

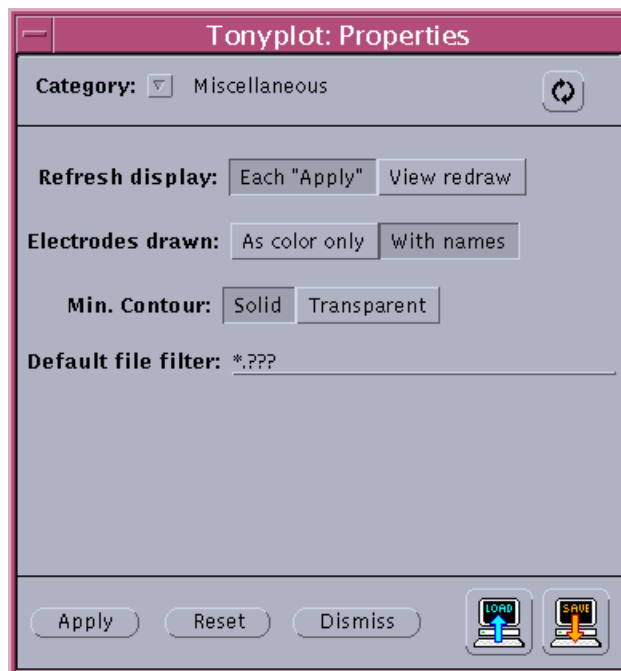
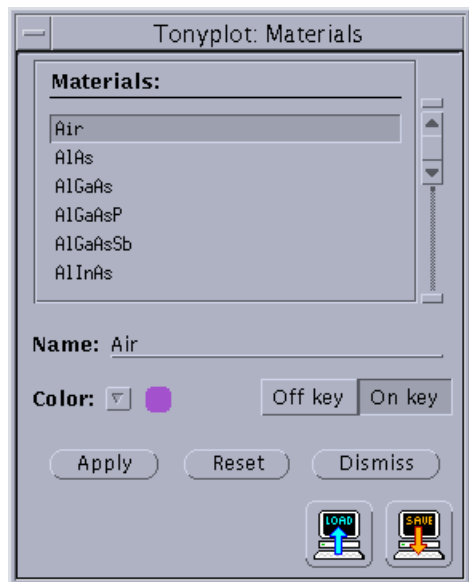


Figure 7-40: Miscellaneous Popup

- **Refresh** — When changing many popups at once, the redraw that accompanies each click of an **Apply** button may not be desirable. If this is the case, set this property to **View redraw**. This means that the view only redraws when **Redraw view** is chosen from the main **View** menu. If this property is set to **Each Apply**, the default behavior is restored.
- **Electrodes drawn** — When electrodes are drawn in Mesh plots, TONYPLOT indicates them by using a hashing in the defined color for electrodes. If this property is set to **With names**, then the names of the electrodes (where defined) are also drawn at a point near to the actual electrode region.
- **Min Contour** — Determines whether the minimum contour is displayed as transparent or not.
- **Default file filter** — Use this to specify the filter used on the **Load structure** popup.

7.19.14: Materials



The colors used by TONYPLOT to represent the different types of materials can be altered. To change the colors, select **Properties**→**Materials** (not from the **Properties** popup). This will open a popup. The items on this popup are:

- **Name list** — The scrolling list shows a list of all material names known to TONYPLOT.
- **Name** — This is the name of the selected material. This cannot be changed.
- **Color** — A palette shows the color currently used for the selected material. Any color can be chosen if the default is not acceptable.
- **Off/On key** — Sometimes it may be desirable not to show a certain material on a material key legend. If this is the case, choose the **Off** key from this item.

7.19.15: Functions

The **Functions** popup (Figure 7-41) can be accessed from the main **Properties** menu, but not from the **Properties** popup. This function and its use to construct custom made expressions is described in the Functions section.

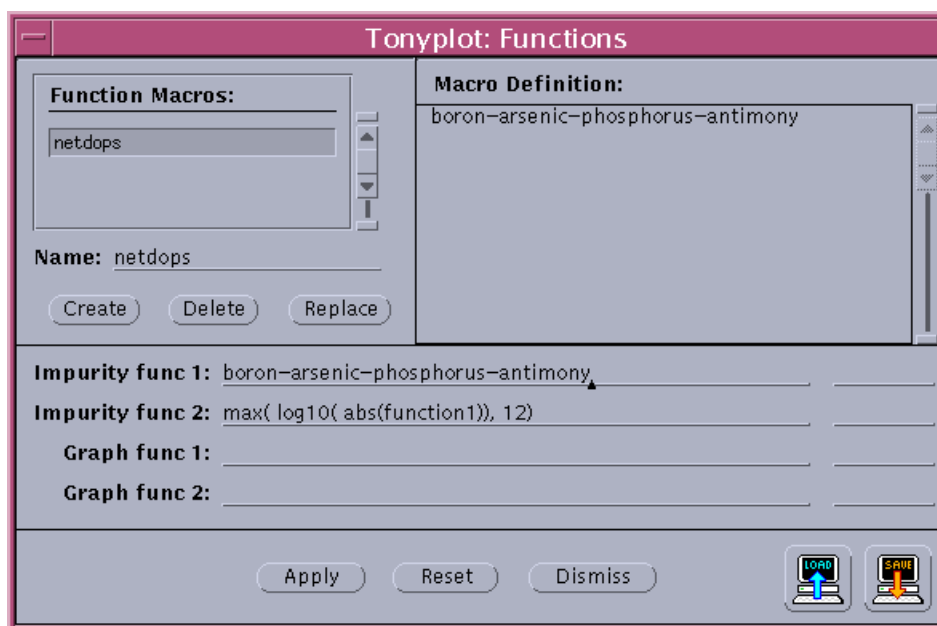


Figure 7-41: Functions Popup

7.20: Key Legends

7.20.1: Overview

Data is of little use without a key to explain its meaning. TONYPLOT uses keys in a variety of situations to explain how the information in a plot is being represented. Each key is drawn in a default position in a plot that does not overlap with any other key. These default positions can be changed, however, if necessary, and you can even “drag” keys to any location in the window. In an overlay plot, a key applies to all levels, since the data displayed on each level is the same. An overlay plot also has its own “level” key, to indicate the different levels in the plot.

7.20.2: Types Of Keys

In an overlay plot, a key applies to all levels, since the data displayed on each level is the same.

The following keys are available:

- **Contours** — A contour plot key indicates the value range for a given color when filled contours are used, and the value of a color if line contours are used. Up to three of these keys may appear at once, since a 2D plot may contain up to three sets of contours. Contour keys show only whole integer values when the quantity is contoured on a log scale.
- **Regions** — Regions can be displayed on 2D mesh plots and Cross section plots. The region key shows region-to-color relationships. Usually, the regions are represented by the materials that they are made of and the key shows these materials and their colors.
- **Graphs** — Whenever graph lines are drawn, there is a graph key. This indicates the different quantities shown by each graph line. These appear, for example, in XY-Graph plots and Cross Section plots.
- **Stats** — A stats key is used whenever a plot of statistical data is present. The exact type key varies according to the manner in which the data is displayed.
- **Vectors** — A vector key indicates the size of arrows and magnitudes that they represent; largest, smallest, and half way between. The magnitude of other arrows can be judged by comparison to this key.
- **Levels** — To distinguish data between levels in an Overlay plot, a level key appears whenever there is more than one level. These show the level-to-color relationship for that plot. By default, overlay levels are distinguished by the name of the file that was used to create each level. However, these names can be changed to any other name specified by the user.

7.20.3: Positioning Key Boxes

There are two ways to position key boxes. First, there is the Key Options category of the Properties popup. This shows a list of all the types of keys mentioned above and an icon along side each one showing its default location. There are six possible default locations: use the choice item and/or the cycle button to choose the ones desired.

The default locations are set up in such a way that there are not any overlapping keys. When choosing new default locations, be aware of the possibility of key boxes overlapping. The settings can be saved as the default settings to be used between sessions.

The second method is to position key boxes manually. To do this, point to a key box on a selected plot, and drag the mouse while holding down the SHIFT key on the keyboard. This allows the key box outline to be dragged to any position in the window. When the mouse button is released, the view is redrawn with the key in the new position.

The position of the key applies to all plots. However, the entire view is not be redrawn when a key has been dragged. The other plots show the key in its new position only when they are redrawn.

7.20.4: Drawing Styles

There are some TONYPLOT properties that control how keys are drawn. These are described in the Properties section, but are outlined again here for reference:

- **Key type** — Can be transparent or opaque. Transparent keys allow the plot underneath to be seen, to some extent, whereas opaque keys overwrite the plot underneath them. Only opaque keys are used in hardcopies.
- **Function label** — When a function is plotted, it can be indicated on the appropriate key with either its name or its definition. For example, suppose function 1 has been set up as $\log(\text{current}/10)$. If **Name** is chosen, the key says **Function 1**, but if **Definition** is chosen, it says $\log(\text{current}/10)$.

7.21: The Command Stream

TONYPLOT supports an input language that is used to control the behavior of the program by using text commands only. This language is called TPCS (for TONYPLOT Command Stream).

It is TPCS that is used in set files. If a set file is examined, it is seen to contain TPCS statements. These statements are read by TONYPLOT and executed, to set the plot display to its state when the set file was created.

There is a close link between TPCS and the TONYPLOT graphical user interface (GUI). Many statements directly reflect actions you perform on the popups (such as the apply statements) and plots (the select statement).

TPCS commands can be used at any time by selecting **Command Stream** from the main **File** menu. Users of csh should make sure that TONYPLOT is running as a foreground process to ensure that the commands typed go to the correct program.

The TPCS prompt is displayed in the window from where TONYPLOT was first started. This looks like this:

```
TPCS>
```

and indicates that TONYPLOT is waiting for commands. At the same time, it is still possible to use the normal GUI to control TONYPLOT.

7.21.1: Help

TPCS has a built-in help system. Enter `help` at the TPCS prompt to get started. Entering `help <word>`, where `<word>` is a topic name (enclosed in quotes) gives a list of commands associated with that topic. For example:

```
TPCS> help "contours"
```

shows a list of all commands that have something to do with contours.

7.21.2: Finishing TPCS

To end a TPCS session, enter an "end of file" character from the keyboard. This will be CTRL-D on most systems.

Syntax

The complete syntax for TPCS is given here. Note that these statements appear in no special order, and no complete explanations are given. To obtain a better understanding of TPCS, you may study set files, or contact Silvaco support for further help.

The following meta notations are used in the syntax descriptions:

<int> — Any integer (i.e., 4).

<string> — Any word(s) enclosed in quotes (i.e., "my world").

| — Indicates a choice of possibilities, each separated by the vertical bar symbol (i.e., on|off is either on or off).

<expr> — A floating point number, or a mathematical expression that evaluates to a floating point number. (i.e., 3.4, 1.3e12, sin(0.1), 4+8/3, etc).

The statements supported in TPCS are as follows:

```
draw <int>
draw all
load <string>
load <string> replace
load <string> overlay
select <int>
select all
select none
select auto
delete
show mesh on|off
show edges on|off
show materials on|off
show contours on|off
show vectors on|off
show light on|off
show junctions on|off
show electrodes on|off
show threed on|off
show points on|off
show lines on|off
contours select <int>
contours impurity <string>
contours type lines
contours type fill
contours outline on|off
contours color <int>
contours maximum auto
contours maximum <expr>
contours minimum auto
contours minimum <expr>
contours nsteps <int>
contours increment <expr>
contours materials all
contours materials <string> on|off
contours apply
threed impurity <string>
threed show edges on|off
threed show yaxis grid on|off
threed show xaxis grid on|off
threed show mesh on|off
threed show grid on|off
threed draw color
threed draw light
threed angle xaxis <int>
threed angle yaxis <int>
threed draw from <int>
threed apply
cutline from <expr>, <expr> to <expr>, <expr>
light beam <int> on|off
light materials all
light materials <string> on|off
light style <int>
light function <int>
```

```
light color <int>
light maximum <int>
light apply
vectors impurity <string>
vectors materials <string> on|off
vectors materials all
vectors maximum <expr>
vectors minimum <expr>
vectors color <int>
vectors apply
mesh2d apply
impurity <string> on|off
impurity none
impurity all
xsection apply
xaxis <string>
yaxis <string> on|off
yaxis log <string> on|off
yaxis none
yaxis all
xygraph type <int>
data type <int>
scale xaxis linear
scale xaxis log
scale yaxis linear
scale yaxis log
scale yaxis all
xygraph apply
title main <string>
title main auto
title sub <string>
title sub auto
show xaxis on|off
show yaxis on|off
show axes on|off
show grid on|off
show label xaxis on|off
show label yaxis on|off
range xaxis <expr>, <expr>
range xaxis auto
range yaxis <expr>, <expr>
range yaxis auto
increment xaxis <expr>
increment yaxis <expr>
label xaxis <string>
label yaxis <string>
annotation apply
zoom from <expr>, <expr> scale <expr>, <expr>
zoom out
zoom previous
label <string> from <expr>, <expr> color <int> scale <int>
label <string> from <expr>, <expr> color <int> scale <int> to <expr>, <expr>
label <string> auto scale <int>
label <int> at <int> <int>
key electrical at <int>
```



```
key electrical at user <expr>, <expr>
key profile at <int>
key profile at user <expr>, <expr>
key contours <int> at <int>
key contours <int> at user <expr>, <expr>
key materials at <int>
key materials at user <expr>, <expr>
key regions at <int>
key regions at user <expr>, <expr>
key vectors at <int>
key vectors at user <expr>, <expr>
key overlay at <int>
key overlay at user <expr>, <expr>
color window <int>
color background <int>
color foreground <int>
color dark border <int>
color light border <int>
color zoom <int>
color grid <int>
color mesh <int>
color edges <int>
color cutline <int>
color junctions <int>
color electrodes <int>
color outline <int>
label overlay <int> <string>
overlay reset
overlay apply
eval <expr>
abs (<expr>)
log (<expr>)
exp (<expr>)
log10 (<expr>)
sqrt (<expr>)
sin (<expr>)
cos (<expr>)
tan (<expr>)
asin (<expr>)
acos (<expr>)
atan (<expr>)
hypot (<expr>, <expr>)
mag (<expr>, <expr>)
sinh (<expr>)
cosh (<expr>)
maximum (<expr>, <expr>)
minimum (<expr>, <expr>)
help help
help all
help <string>
quit
comment
```

7.22: Functions

Functions are included in TONYPLOT to allow you to further customize the output that can be produced, and to extend the amount of data that can be plotted without needing further simulation runs and large data files. TONYPLOT allows functions to be created for use with any type of plot, and uses an advanced mathematical parser to calculate function results from arbitrarily complex math expressions.

7.22.1: Use Of Functions

In order to use functions, first define a function, in terms of quantity names (Boron, Drain bias, Temperature etc), constants, and operators (sin, sqrt, +, — etc), and then PLOT that function in the desired plot. Using scientific notation in functions (i.e., 1e23) requires that the mantissa have a decimal point (i.e., 1.0e23), otherwise it does not work.

The **Functions** popup is used to define functions. It can be displayed from the main **Properties** menu, or from buttons marked **Functions...** that appear on each of the three types of **Display** popup.

Once defined, a function is plotted by choosing the name of the function (Function 1 or Function 2) from the relevant control item on the **Display** popup.

7.22.2: Defining Functions

The **Functions** popup is split into two sections: the top section allows function macros to be set up, and the lower section is where the functions are actually defined. Note that there are two types of function:

- **Graph functions** — These are used on XYGraph plots only and are defined in terms of graph (electrical) quantities.
- **Impurity functions** — These functions can be plotted in Mesh plots and Cross Section plots only, and are defined in terms of impurities.

Two of each type of function is supplied and either or both can be shown on any plot. It is possible to nest functions, by including the name of one function as a variable in another.

7.22.3: Plotting

When **Function 1** or **Function 2** are chosen to be plotted, TONYPLOT evaluates the results of the function at each data point and stores these values in the data attached to each plot level. Then the function can be drawn, along with any other quantity also selected.

Example

Suppose a Master file contains values for the four dopant impurities boron, arsenic, phosphorus and antimony. We can use a function to compute the net doping by entering the following as Function 1:

```
boron - arsenic -phosphorus - antimony
```

However, most plots of net doping are shown on log scales (this is the default for TONYPLOT) so you need to calculate the log of this sum. Make sure the total is positive beforehand, so use `abs()` to get the absolute value:

```
log10 (abs (boron - arsenic - phosphorus - antimony))
```

The data for each dopant, however, is not useful below values of around 1e12. TONYPLOT usually does not show values below this level, but with a function it cannot tell that this is needed. Hence we use the `MAX()` operator to keep the function result within a useful range:

```
max (log10 (abs (boron - arsenic - phosphorus - antimony)), 12)
```

Simplify this expression by splitting into two functions, and nesting one inside the other, as follows:

```
Function 1 = boron - arsenic - phosphorous - antimony
```

```
Function 2 = max (log10 (abs (Function 1)), 12)
```

Now plot “Function 2” on a Mesh or Cross Section plot, it shows the profile of Net Doping (clipped at $1e12$). The Function popup used to define this function is shown in Figure 7-42.

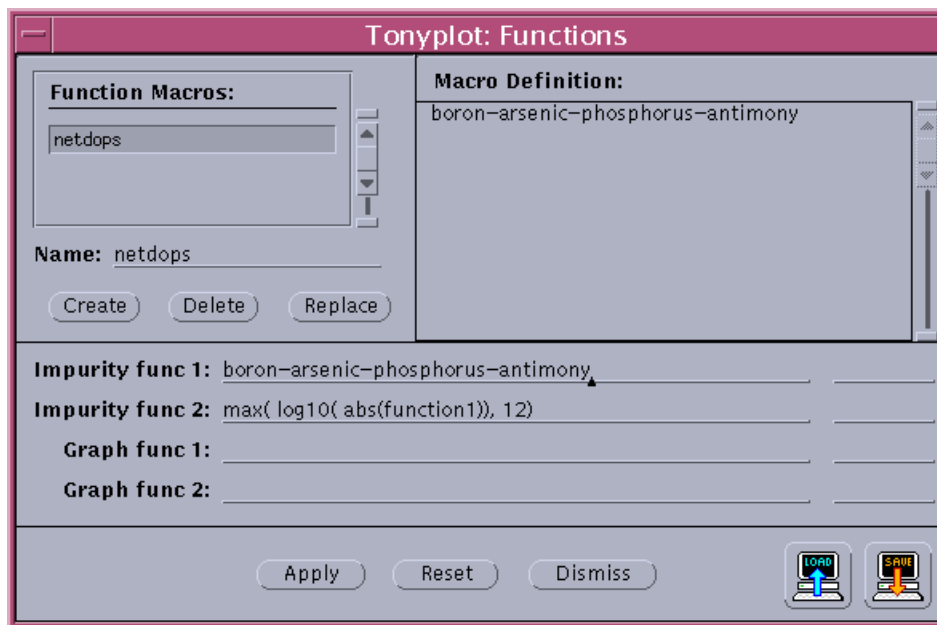


Figure 7-42: Functions Popup

7.22.4: Function Macros

To allow further simplification of functions, the macro section is provided on the Functions popup. It can be used to store common functions and identify them with an easy to remember name. The items used to manipulate macros are as follows:

- **Macro List** — This shows all the macro names currently known to TONYPLOT. Select names here to view or edit the macros.
- **Name** — Shows the name of the selected macro, and is used to change macro names and add new macros.
- **Definition** — This is a small edit window where the full macro definition is displayed. New definitions should also be entered using this edit window.
- **Create** — To create a new macro, enter a new name and definition for the macro, and click on this button. The new name appears in the list. If the name already exists, the old definition is replaced.
- **Delete** — This deletes the macro that is currently selected into the name list.
- **Replace** — This can be used to change a macro name and/or definition. Enter the new text and click on this button to replace the selected macro.

The macro names can then be used in any function just as though the whole definition had been typed.

For example, in the “Net doping” example, we could add a macro called “net_dop” and in the definition window enter:

```
max (log10 (abs (boron - arsenic - phosphorus - antimony)), 12)
```

Then we could define either **Impurity Function** to simply be net_dop

This makes the function definitions easier to read, and also allows useful names to be added to the plot key. The Property called “Function label” can be used to change how functions are labelled on the plot key.

All macros can be saved to a defaults file for use any time TonyPlot is used. Two buttons on the popup allow defaults to be saved and loaded at any time.

7.22.5: Function Syntax

Functions are constructed just like normal math expressions, but with names of quantities from data files used as variable names. Functions can be built with the following operators:

$a + b$	a plus b
$a - b$	a minus b
a / b	a divided by b
$a * b$	a multiplied by b
$a ^ b$	a to the power b
-a	negative a
abs(a)	absolute value of a
log(a)	natural log (base e) of a
exp(a)	inverse natural log of a (e^a)
log10(a)	log base 10 of a
sqrt(a)	square root of a
sin(a)	sine of a (a in radians)
cos(a)	cosine of a (a in radians)
tan(a)	tangent of a (a in radians)
asin(a)	arcsine of a
acos(a)	arccosine of a
atan(a)	arctangent of a
sinh(a)	hyperbolic sine of a
cosh(a)	hyperbolic cosine of a
mag(a,b)	magnitude of vector [a,b]
hypot(a,b)	hypotenuse of a and b (same as mag(a,b))
max(a,b)	maximum of a and b
min(a,b)	minimum of a and b
dydx (a,b)	derivative of a with respect to b

Normal operator precedence is obeyed. Expressions can use parentheses to change the operator precedence where needed.

If an expression contains an error, this is reported when the function is plotted. Invalid values are for the most part plotted as zero, except for log() which uses the value predefined in the Property called “Log Zero”. If a function does contain an error, TONYPLOT displays a notice box informing you of the type of error encountered. A syntax error causes all function values to be zero. An evaluation error causes just the offending data points to be zero.

The derivative function dydx can take any two variables. It also accepts “distance” to represent the x value, i.e., for a vertical cutline dydx (boron, distance) would give the derivative of boron concentration against depth.

7.22.6: Function In TPCS

Expressions can be evaluated in TPCS, but TPCS does not perform the variable substitution the functions perform. To evaluate an expression in TPCS use the following command:

```
TPCS> eval <expression>
```

where <expression> is constructed according to the syntax rules explained above. The result is printed out below the input. For example:

```
TPCS> eval log10 (sqrt(sin(0.4)))  
-0.204792  
TPCS>
```

When TONYPLOT plots functions, it uses TPCS commands such as these to work out the new values. Hence this can be used to check for computation or syntax errors, or even for a quick calculator.

7.23: User Data Files

7.23.1: Overview

User data files are ASCII text files that can be loaded into TONYPLOT, and have a easy-to-read format. This means that data from external sources can easily be read into TONYPLOT. Those who have their own sets of data already, maybe from other software packages or from experimental results, find this format useful, since it allows the features of TONYPLOT to be applied to that data.

Data files can be compared to other files types as well. This makes it simple to compare simulation results from Silvaco simulators with data obtained from real-life experiments.

7.23.2: Loading User Data Files

TONYPLOT automatically recognizes data files in this format, and so it not necessary to use any special command line options to load them. They can also be loaded using the File Loader popup, just like regular Silvaco files.

7.23.3: Creating User Data Files

Data files can be created in any manner. From other programs, filters, or TONYPLOT features, such as **Export**. If needed, you can also edit the files.

7.23.4: Data Format

Data files should be constructed in the following manner.

```
<title line>
<r> <c> <t>
<title 1>
<title 2>
...
<title t>
<x11> <x12> ..... <x1c>
<x21> <x12> ..... <x2c>
<x31> <x12> ..... <x3c>
... ..
<xr1> <xr2> ..... <xrc>
```

title line can be any sequence of ASCII characters. This is used as the main plot title, and must be present.

r is the number of rows in the file. This must be present, but can be given as zero, in which case TONYPLOT works out how many rows there are.

c is the number of columns. This must be present, but can be given as zero, in which case TONYPLOT works out how many columns there are.

t is the number of column titles. If not present, it defaults to zero (i.e., no titles).

title X is the title that is assigned to the data in column number X. These titles are optional, but if given, the number of titles must match the t parameter in the second line.

xij is a data value, in row i, column j.

Each column in the file represents data values for a certain quantity. if no titles are given, these quantities assume default names of "User data 01" for the first column, "User data 02" for the second, and so one. The titles allow more meaningful names to be added.

7.23.5: Examples

The following are examples of some user data files, and descriptions of each. These can be entered into a text file and tested with TONYPLOT, if needed:

Equation

This simple example plots a section of a curve of the equation $y = \sin(x)$. This type of file is the simplest.

```
y = sin (x)
10 2
0.1 0.09983
0.2 0.19867
0.3 0.29552
0.4 0.38942
0.5 0.47943
0.6 0.56464
0.7 0.64422
0.8 0.71736
0.9 0.78333
1.0 0.84147
```

Transistor

This example contains information about an NMOS transistor. This example shows how columns are named.

```
1.0 um NMOS Id/Vg
10 2 2
gate bias (V)
drain current (A)
0.0 1.0E-14
0.1 1.0E-13
0.2 1.0E-12
0.3 1.0E-11
0.4 1.0E-10
0.5 1.0E-09
0.6 1.0E-08
0.7 1.0E-07
0.8 5.0E-07
0.9 8.0E-07
1.0 8.9E-07
```

Display

User data files are treated in the same way as normal XY Graph plots in TONYPLOT. This display popup for these plots is exactly the same as the Graph display popup. In fact, once loaded into TONYPLOT, there is no difference between these two type of data at all.

7.24: Set Files

When a plot has been set up so that it displays the desired set of information (by use of the display popups, labels, annotation and so on) it is possible to save this information into a set file so that the display can be recreated automatically by TONYPLOT. A set file contains instructions that tell TONYPLOT the steps needed to recreate the same display that was visible at the time the set file was created. Thus, when the same data files are loaded into TONYPLOT at a later date, or into a different TONYPLOT, loading the set file avoids having to go through all the popups again.

By convention, set files usually have a suffix of `.set` which denotes them as set files. The Set File Loader uses a `.set` default filter.

7.24.1: Creating

To create a set file, the **Set Files** popup is used, accessed from the main **File** menu. This popup looks similar to the **File Loader** except that it has both a **Load** and a **Save** button. Move to the directory where the set file is to be created, and enter the name of the desired file into the field marked File name. If the file already exists, select it from the scrolling list (use the **Filter** to screen out undesired files from the list). When the **Save** button is clicked on, the set file is created. Confirmation is required if the file is overwritten.

7.24.2: Loading

To load a previously created set file, locate the file using the **Set Files** popup and click on the **Load** button. After a short while the view is updated. Additionally, set files can be loaded from the command line with the `-set` option. If an error occurs when loading the file, a warning notice appears.

When a set file is created while there are multiple plots in the view, it is important that the same plots are present when the set file is loaded, since a set file cannot store information about duplicated or deleted plots. For example, suppose TONYPLOT is started with the name of one data file:

```
% tonyplot diode.str
```

When the plot appears, you can show contours in one window and the mesh in another. The plot is duplicated to allow this, and the appropriate display parameters applied to each plot. Then a set file is saved, called (for example) `setup.set`. You then quit from TONYPLOT.

At a later date, if the same view is to be created automatically, you must enter:

```
% tonyplot diode.str diode.str -set setup.set
```

because there were two plots of `diode.str` when the set file was created. If only one file name were given, the set file would not be loaded completely.

Similarly, when plots are deleted, restart TONYPLOT only with the names of files that were being plotted when the set file was created.

Set files cannot record any actions you performed to create overlay plots, although they can store setup information about the overlay itself. Therefore, you must recreate the overlay manually before the set file is loaded. For example, suppose TONYPLOT is started with two data files as follows:

```
% tonyplot drn.log src.log
```

and you then overlay these two files. Also, assume you delete the second plot, i.e., the plot of `src.log`, leaving a plot of `drn.log`, and a plot of both data sets overlaid. Then you saved a set file, `log.set`. To recreate this view in another TONYPLOT window, you must enter:

```
% tonyplot drn.log drn.log -overlay src.log -set log.set
```

This also illustrates the point about not including filenames from deleted plots when restarting.

Note: The deleted src.log plot was not included in the command to restart TonyPlot.

Cross section plots that are generated using the **Cutline** tool can be created from the original 2D Mesh plot using set files. Hence, it is possible to save a view with a mesh and a cross section and recreate it from the mesh alone. For example, suppose a user loads a mesh file called `pmos.str`, creates a cutline cross section, and then saves a set file called `cut.set`. The view can be recreated with:

```
% tonyplot pmos.str -set cut.set
```

The set file automatically takes a cutline from the PMOS mesh and display the cross section plot as it appeared when the set file was created.

7.24.3: Setfile Syntax

Set files use TPCS statements to store setup information. Also included in a set file is the version of TPCS used to create the file. Those familiar with TPCS commands will be able to create set files by hand where necessary, or modify existing set files. Set files can also be used as examples of various TPCS syntax rules.

7.25: Overlays

One of TONYPLOT's most useful features is the ability to directly compare different sets of structure data. This not only means loading several output files at once and looking at each one in the same view, but also overlaying the data sets in the same plot subwindow.

When different files are overlaid, the plot is called an overlay plot, and has some characteristics not shown by normal plots. Each structure is drawn on a separate level in an overlay plot, and the levels are transparently stacked upon one another. Each level uses the same axes and display settings, so that the data can be readily compared. Obviously, each level must be of the same plot type (2D Mesh, Cross Section or XY Graph).

7.25.1: Making An Overlay

Overlay plots can be created in one of two ways. If the files to be overlaid are already loaded into TONYPLOT (each in its own plot subwindow), a new overlay plot can be created by selecting the plots to be overlaid (at least two must be selected), then choosing **Make Overlay** from the main **View** menu. A new plot is created containing one level for each plot selected.

Alternatively, if data that is to be overlaid is not loaded, the new files can be overlaid onto an existing plot as they are loaded. Use the **Overlay** option from the **File Loader** popup to do this, or the `-overlay` option if loading from the command line.

7.25.2: Splitting An Overlay

An overlay plot can be broken down into separate single-level plots by choosing the Split overlay option from the main View menu. The overlay plot to be split must be selected when this option is chosen. One new plot will be created for each level in the overlay.

7.25.3: Overlay Control

Overlay plots are controlled just like any other plot. It is still a single plot even though several structures are displayed within it. Zooming, key commands, labeling etc. are still possible as though only one structure were present.

7.25.4: Overlay Display

Plot display for an overlay plot is exactly the same as for a normal plot, except that multiple data sets are affected. All the data sets (i.e., every level) is displayed according to one common display setting. It is not possible, for example, to show contours in one level and vectors in another. Since each level must be of the same plot type, the popup used to change the display settings are the same as described for normal plots.

Some levels may not be plotted if the display settings includes quantities or options that are not present in the data for that level. The display popups, however, shows all quantities from all levels. For example, one level may contain "boron" and another level just "arsenic". The display popup shows both "boron" and "arsenic", but if only "boron" is chosen, only the first level is drawn.

7.25.5: Identifying Data

Data from separate levels can be identified by the overlay key. This key indicates which profile or graph line corresponds to a certain data file. The quantities plotted are identified with the same key used in single-level plots.

Each level is shown in a different color, (line type for monochrome screens) and each quantity with a different mark type. The colors used are determined by the current set of "Sequence colors". See the Properties section to determine how to set these colors.

Mesh plots, when overlaid, use the same color for each level. Because of the large amounts of information that can be portrayed in a 2D Mesh plot, the number of levels is limited to three. For **XYGraph** and **XSection** plots, the limit is practically unlimited.

7.25.6: Level Names

Each level in an overlay is named from the file from which its data was taken. If needed, these names can be changed with the **Level names** popup. Choose **Level names...** from the **Plot** menu to display this popup.

To change the name of a level, select the old name from the scrolling list, and enter a new name into the textfield labeled **Name:**, and press the Return key (do not forget to press Return, or else the new name will not be stored). Repeat this for each that is to be changed, and click on the **Apply** button. The plot updates to show the new names.

7.25.7: Cutlines

When the **Cutline** tool is used on an overlay 2D Mesh plot, a section is taken from each level. TONYPLOT automatically overlays each of these when it creates the new cross section plot.

7.25.8: Properties

There are some TONYPLOT properties that apply to overlays. These are accessed by choosing **Overlays** from the main **Properties** menu or **Properties** popup. Although explained in the Properties section, they are given here again for reference.

Creating — When creating overlays from existing plots, TONYPLOT just creates a new plot. The old plots still remain in the view. If the **Deletes originals** option is chosen however, the plots that made up the overlay are deleted when the overlay is created.

Level colors — As explained above, each level is identified with its own color. If the same color is to be used for all levels, choose **Single** for this property.

No. of levels — As explained previously, the number of levels in a plot is limited. However, it is possible to override this limit and have as many levels as required in an overlay. Use this switch to turn this limit on or off.

Display option — Although the default setup is to use different colors for different levels, and different mark symbols for different quantities, this can be reversed by using the **Display** option property. Select **color/mark** or **mark/color** as desired.

7.26: Production Mode

7.26.1: Outline

TONYPLOT provides the graphics behind the VWF PRODUCTION MODE, using a special set of controls and popups. With **Production** mode enabled, it is possible to examine and interact with response surface models (or RSMs) in one, two or three dimensions. These RSMs can be examined with a selection of PRODUCTION MODE features such as **Failure Analysis**, **Disposition**, and **Synthesis**.

An RSM consists of a “response parameter”, sometimes called an “output” or simply “model”, and a number of “input parameters”. The output is calculated from the inputs according to the RSM definition, which is passed from VWF to TONYPLOT.

RSM plots are displayed as either simple XY graphs, where the model is plotted against the variation of one input, or as 2D or 3D contour plots, where the model is plotted against two inputs. In all cases, inputs not plotted are held at fixed values, although you can set these fixed values.

Control of RSM plots is described in the Plot Control section. See those pages for an explanation of how to draw RSM graphs, contours and surface plots.

This section discusses the advanced PRODUCTION MODE features available.

7.26.2: Enabling Production Mode

There are three ways to enable the VWF PRODUCTION MODE graphics in TONYPLOT. The simplest way is to let TONYPLOT do it automatically: whenever an RSM plot is loaded, PRODUCTION MODE is started. The second way is to use the `-production` command line argument when starting TONYPLOT. The third method is to choose PRODUCTION MODE from the main **File** menu.

When PRODUCTION MODE is started, there is an extra menu along the top of the main TONYPLOT window. The new menu is labeled **Production** and appears between the **Print** and **Properties** menus.

This menu allows access to each of the five PRODUCTION MODE features (Interactive control, Failure Analysis, Calibration, Synthesis and Yield Analysis) as well as some parameter editing popups (i.e., Input range, SPC limits). Choosing any of the main features displays the Production popup, in the appropriate mode. This popup is explained in detail later. Selecting one of the editing popups displays that popup.

7.26.3: The Production Mode Popup

By selecting an option from the main **Production** menu, the **Production Mode** popup appears. This popup shows all the RSM input and output parameters currently in use, and allows the selection of some advanced features.

The popup can be set to one of five “modes” (Interactive, Failure Analysis, Calibration, Synthesis and Yield Analysis) by either choosing the mode from the **Production** menu, or from the choice item at the top of the popup.

Below the mode selector is a panel whose contents depend on the mode currently selected. In the Interactive mode, for example, it shows the current **Process Name** (if any) and a row of four buttons for control of the input sliders.

The input sliders appear on the next panel down. Each input parameter in use is represented as a slider/toggle combination. Inputs can be “selected” by pressing the toggle — it is grayed out when not activated.

The popup can show up to twenty four inputs. The popup can be expanded or reduced to show more or fewer sliders with the “plus” and “minus” buttons found in the lower right corner. TONYPLOT tries to size the popup so that all inputs are shown, and unused sliders are hidden.

At the bottom of the popup are the usual **Reset** and **Dismiss** buttons, and also a **Define** menu button. This allows access to some set-up popups, which will be explained later.

7.26.4: Interactive RSM Control

The Interactive mode is the basic use for the **Production** popup. When set to this mode, the mode control panel shows the current **Process Name**. This name can be changed if needed.

Under the process name are four buttons that control the input parameter sliders:

- **Reset to nominal** — Clicking on this button sets all slider positions to the “nominal value” for each input. The nominal value is defined by the RSMs, but can be changed to any other value with the **Input Parameter Ranges** popup, described later, or with the next button.
- **Store as nominals** — When this button is clicked on, the current position of all selected input sliders is used as a new nominal value for those inputs. Nominal values can also be set from the **Input Parameter Ranges** popup, described later.
- **Fix Y Axis** — When you toggle this option, the y-axis ranges of the selected RSM plots remain when you move the input sliders.

7.26.5: The Input Sliders

Any input slider that is selected (i.e., the toggle is green) can be dragged left and right to interactively change the current value of that input.

As the value changes, any selected RSM plots in the current view updates to reflect these new values. Any selected RSM plot is updated as follows:

- Any 1D RSM plot that has the input being plotted on the X-axis moves the gunsight along the curve to match the changing X-axis value. If the input is not the X (or Y) quantity, the whole curve will be recomputed for the new value.
- Any 2D plot that doesn't have the input as either the X- or Y-axis quantity is redrawn, with the contours recalculated for the new input value.
- Any 3D plot that doesn't have the input as either the X- or Y-axis quantity is redrawn, with the contours and surface recalculated for the new input value.

Note: Since plots are updated interactively, it is usually preferable to set the RSM plot density to “low”. This speeds up the calculations and greatly improves the response time.

This interactive control is available regardless of the current popup mode. That is, sliders can be dragged in **Failure Analysis**, **Synthesis**, or any other mode, and plots are still updated interactively, as described above.

7.26.6: Failure Analysis

Description

The **Failure Analysis** feature can be used to predict which input parameter is the most likely cause of an error in production, i.e., an output parameter or measurement that exceeds the specification or design limits.

Set up

When the **Production** popup is set for Failure Analysis, there is a list of all output parameters, a text field for entry of “failed values” and **START** and **STOP** buttons. There is also a **Method** item.

- **Failed Values** — For each output parameter, enter the failed value that was measured at the production site. Press the Return key to submit this value to the list.

- **Method** — This controls the error tolerance method used in the failure analysis. Higher orders are less error tolerant, lower orders are more tolerant.
- **Select Inputs** — From the set of input sliders, choose the ones to be used in the analysis. TONYPLOT tries to find which of these inputs is the most likely cause of failure in all outputs. At least two inputs must be selected, because the algorithm compares relative probabilities of failure cause.
- **START** — Click on this button to start the failure analysis. As each of the selected input parameters are tested, its slider moves from the minimum to the maximum value. Once all inputs have been tested, a plot of the results appears.
- **STOP** — Click on this button to stop the analysis at any time. All calculations are aborted and no result plot appears.

The Result Plot

When the failure analysis has been run, a barchart appears (which is a form of the general TONYPLOT “Stat” plot). The barchart shows the relative probabilities of each input being the cause of the failed values.

Note: Each input is tested individually. The analysis tries to find the one single input that could cause the specified failure in all outputs. The highest bars show inputs that are most likely to have caused the failure(s).

TONYPLOT also adds a set of labels to the plot. These labels indicate the values of each input which would have generated the failed value, if that input had been the cause of the failure.

7.26.7: Synthesis

In **Synthesis** mode, TONYPLOT tries to find a value for each selected input that produces a desired set of outputs. Two algorithms are provided to perform this reverse calculation — a “Levenburg-Marquardt” optimizer and a method known as Adaptive Simulated Annealing.

Setup

Before starting the synthesis calculations, the desired output values must be set. Use the text field labeled **Target Value:** to enter a desired value for the output currently selected in the list on the left. Press the Return key to submit this value.

When all target values are entered correctly, select the desired method. There is a choice between the **Levenburg-Marquardt** optimizer, and **Adaptive Simulated Annealing**. Select the method desired.

Select the input sliders that are to be used in the synthesis: not all of them need to be used. Unselected inputs will be fixed at their current values when the RSM computations use them.

Certain operational parameters specific to each synthesis method can be customized to help obtain the required results. See the Optimizer Setup or ASA Setup sections for further details.

Click on the **START** button to start the synthesis procedure. As the calculations progress, the latest results achieved will be displayed in the **Current value** text field. A status message describes the current state.

To abort the synthesis at any point, click on the **STOP** button. The procedure is cancelled and reset.

Results

The inputs sliders changes as the synthesis calculations progress. When the procedure is complete, and it succeeded, the input slider positions shows the values of the inputs needed to produce the output value(s) that were specified in the list of targets.

7.26.8: Yield Analysis

Description

Yield Analysis is a prediction tool used to simulate yield in a real-life fab. TONYPLOT generates large numbers of output measurements from a statistical sample of many inputs. By specifying the probability distributions of each input parameter, a total distribution of all outputs can be obtained.

The statistical distributions of input parameters are obtained in several ways. First, they can be passed to TONYPLOT along with the actual RSM data that uses the inputs. If none are passed, TONYPLOT generates a default distribution. Finally, all input distributions can be specified using the **Input Distributions** popup (see the Input Distributions section).

Setup

The first thing to set up before running a **Yield Analysis** is the input distributions. The scrolling list on the **Production** popup shows the distribution types for each input.

To examine the distributions in detail, and possibly alter them, use the **Input Distributions** popup.

The number of samples taken by TONYPLOT of each input parameter can be set to any value; the default is 5000 samples. The more samples that are taken, the more accurate the results, but it takes longer to perform the analysis.

Click on the green **START** button to start the analysis. The number of samples taken is continuously displayed next to the start and stop buttons. To abort the analysis at any time, click on the **STOP** button.

When all samples have been taken, TONYPLOT takes a few moments to collate the data it has obtained, and then a barchart is plotted.

Results

The default result plot that Yield Analysis creates is a barchart showing the distribution of the output parameter(s). This plot is an instance of a standard **Statistics** plot, and can be manipulated further with the normal popups for **Statistics** plots.

The plot also contains data for all the input values that were used: these can also be plotted. Use the Stat plot **Display** popup to select sets of data to be plotted, and the type of plot. It is possible to draw pie charts, scattergrams, box plots, and other figures, to examine the data. See the Statistics Display section for more information.

7.26.9: Input Parameter Ranges

As RSM data is loaded into TONYPLOT, a record is kept of the greatest range of each input parameter. Along with these minimum and maximum values, a “nominal” value is stored, which represents the normal, or default, value of that input.

It is possible to alter the range and/or nominal value of any input by using the **Input Parameter Ranges** popup, which can be displayed by choosing this option from the **Define** menu, found at the bottom of the **Production** popup.

Control Items

The popup shows a list of all inputs from all loaded RSMs, and shows the minimum, maximum, and nominal values for each one. Underneath the list, the name and values for the input currently selected are displayed: the values can be changed, but the name cannot. Press the Return key on each text field to submit new values to the list. TONYPLOT makes sure that the values are logical, i.e., that the minimum is less than the maximum and the nominal lies between them.

Once all the desired changes have been made to the list, it is necessary to click on the **Apply** button, which submits these new values to TONYPLOT. Any RSM plots are updated to use these new ranges and nominal values.

The **Reset** button resets the popup to state it was in when the **Apply** button was last clicked. In other words, if changes are made to the list that are not correct or are not needed, click on Reset to put all values back to the way they were when the popup first appeared.

If the user-defined values are to be completely discarded, click on the **Restore** button. This sets all values back to the “default” settings, i.e., ranges and nominals are taken from the RSM data sets, and user-defined changes are cancelled.

Uses

The **Input Parameter Ranges** popup affects the range over which values are taken for each input when RSM outputs are computed. The sliders on the **Production** popup reflect the current range for each input.

Some models may only be valid for specific ranges of their inputs. Since it is possible to exceed these ranges, there is a feature which allows the valid range to be identified on a 1D plot (see the RSM Display section).

7.26.10: Input Distributions

As RSMs are loaded into TONYPLOT, a default distribution is assigned to each one. This distribution represents the statistical “spread” of values that would be obtained for this input parameter in an experimental situation. Using this data, TONYPLOT can simulate real-life input values by sampling data with the given distribution parameters.

The default distribution given to each input is Gaussian, with a mean value half way between the minimum and maximum value of that input. The standard deviation will be 10% of the mean.

To alter the distributions for any input, use the **Input Distribution** popup, which is displayed when this option is chosen from the **Define** menu at the bottom of the **Production** popup.

Control Items

The **Input Distribution** popup shows a list of all inputs currently loaded, and the type of distribution for each one, along with its mean and standard deviation.

Underneath the list, the values for the currently selected input are shown. The name of the input is fixed and cannot be changed, but the other items can. Press the Return key on the **Mean** and **Std.Dev** text fields to submit changes to the list. If a new distribution type is selected, the list is updated automatically.

When all desired changes have been made, click on **Apply** to commit the new values. If the **Production** popup is set to **Yield Analysis** mode, the new distribution types are shown in the scrolling list.

To examine and alter the distributions graphically, press the **View...** button. This summons a small window showing a histogram distribution of the selected parameter. Select other parameters to view their distributions also. The sliders under the histogram can be used to modify the mean and standard deviation of the parameter. Click on **Apply** to save the values back to the original scrolling list.

Clicking on the **Reset** button discards any changes made to the popup since it was summoned (or the **Apply** button was clicked).

Uses

Input distributions are used in **Yield Analysis**. When TONYPLOT takes input samples, it uses these distribution parameters to generate realistic values that may be found in a real-life scenario.

For accurate **Yield Analysis** results, each input distribution should be set up so that it matches as closely as possible that found by experimentation.

7.26.11: SPC Limits

Each output parameter used in **Production** mode has a set of “Statistical Process Control” (SPC) parameters, which can be used to monitor the value of some measured value. There are five SPC limits: upper and lower spec limits (maximum and minimum values permitted), upper and lower control limits (ideal maximum and minimum), and center limit (ideal value), which are abbreviated to USL, LSL, UCL, LCL, and CL respectively.

Values for each of these are sometimes passed to TONYPLOT through the RSM data. If not, or if they need to be modified, the **SPC Limits** popup can be used to add or modify **SPC Limit** values for any output, and it can be displayed by choosing **SPC Limits** from the **Production** popup **Define** menu.

Control Items

The scrolling list on the SPC Limits popup shows all output parameters and the USL, UCL, CL, LCL, and LSL values for each one. Underneath the list are five text fields where these values can be changed.

Use the scrolling list to select the output that is to be modified, and enter new SPC values into text fields provided. Press the Return key in each text field to submit changes to the list.

If an output parameter has no defined SPC limits, use the word **None** to indicate missing values. To remove defined values, enter the word **None** into the text field and click on the **Return** key.

When all desired changes have been made, click on the **Apply** button to store the new values. Values in the list are not stored until the **Apply** button is clicked.

Uses

SPC Limits are used to monitor measured output values, to ensure that these values stay within predefined boundaries and generate some warning when the boundaries are crossed.

Outputs generated by RSMs in TONYPLOT can be compared to SPC Limits in a similar way. As long as these values are defined, the control lines can be added to any 1D RSM plot. See RSM Display for details on how to add these SPC limits to a plot.

7.26.12: Experimental Results

Each output parameter modeled by an RSM, has an associated experimental value, that was measured when the process input parameters used in the model were set to their nominal values. If no experimental value is given for an output, or one needs to be changed, choose **Experimental Results** from the **Production** popup’s **Define** menu to display this popup.

Control Items

The popup shows a scrolling list with a line for each output. Next to the name of each output are four values: Experimental result, Modeled result, Error delta, and Predicted value.

- **Experimental result** — This is the measured value, which can be set from information provided by the RSM as it is passed from the VWF, or which can be modified or added later using the text field underneath the scrolling list.
- **Modeled result** — As previously mentioned, the experimental result is measured when all inputs are set to their nominal values. The modeled result shows the value that is calculated by the RSM,

and is a measure of the accuracy of the model. It also used to make a first-order error correction of the model.

- **Error delta** — The error delta is the difference between the experimental measured value and the modeled value when all inputs are nominal. The smaller this error, the more accurate the model. This delta is used to error-correct the model.
- **Predicted value** — This shows the modeled result after error-correction has been applied. It is the same as the experimental results, to show that after error-correction, the model is more accurate (when the inputs are nominal).

To change the experimental result for any output, select the appropriate row from the list, enter the new value in to the text field provided, and press the Return key. Once all new values have been made, click on the **Apply** button to commit the changes.

Uses

A measured result for an output parameter allows TONYPLOT to make a simple error correction to a model of that parameter. The value shown as the “error delta” is added to values obtained from a model to “shift” the response curve to a more accurate position.

The nominal case is used for this calibration (i.e., when all inputs are at their nominal vales), and the result applied for all modeled values.

7.26.13: Optimizer Setup

The **Synthesis** mode of the **Production** popup provides two synthesis methods: the Levenburg-Marquadt optimizer is one of these choices, and some parameters specific to this method can be customized as required.

To access the parameters, choose **Optimizer Setup** from the **Define** menu found on the **Production** popup. A popup appears, allowing customization of these parameters (default values shown in parentheses):

Max. no of iterations (100)
Norm of gradient bound (1e-6)
Sum of squares difference (1e-11)
Marquadt initial value (0.1)
Marquadt scale factor (2)
Marquadt upper bound (1000)
Switch (0.1)
Maximum Jacobian (100)
Maximum RMS error (1)
Maximum average error (1)
Maximum maximum error (1)
Maximum iterations (4)
Noise level (1e-18)
Error ceiling (100)

When all changes have been made, click on the **Apply** button to store them. Click on **Reset** to set the values back to their previous settings.

7.26.14: ASA Setup

The **Synthesis** mode of the **Production** popup provides two synthesis methods: Adaptive Simulated Annealing is one of these choices, and some parameters specific to this method can be customised as required.

To access the parameters, choose **ASA Setup** from the **Define** menu found on the **Production** popup. A popup appears, allowing customization of these parameters (default values shown in parentheses):

- **Limit acceptances (10000)**
- **Limit generated (99999)**
- **Cost precision (5%)**
- **Maximum cost repeat (1)**
- **Number cost samples (5)**
- **Cost parameter scale (1)**
- **Temperature anneal scale (100)**
- **Testing frequency modulus (5)**

When all changes have been made, click on the **Apply** button to store them. Click on **Reset** to set the values back to their previous settings.

For a complete description of ASA its terms and how it is used, see the ASA manual.

ASA code is made available under terms of the GNU general public license for libraries. See the file `$SILVACO/etc/GNU` for details about this license.

This page is intentionally left blank.

8.1: Overview

TONYPLOT3D is a 3D graphics viewer, capable of displaying data generated from 3D process and device simulators. It also allows you to control labeling, lighting, shading and other plot aspects.

This chapter will explain its applications, assuming TONYPLOT3D is properly installed. More detailed information on installing TONYPLOT3D and proper operation on each platform can be found in the Section 8.7: “Operating Platforms”. Figure 8-1 shows TONYPLOT3D when it’s first loaded.

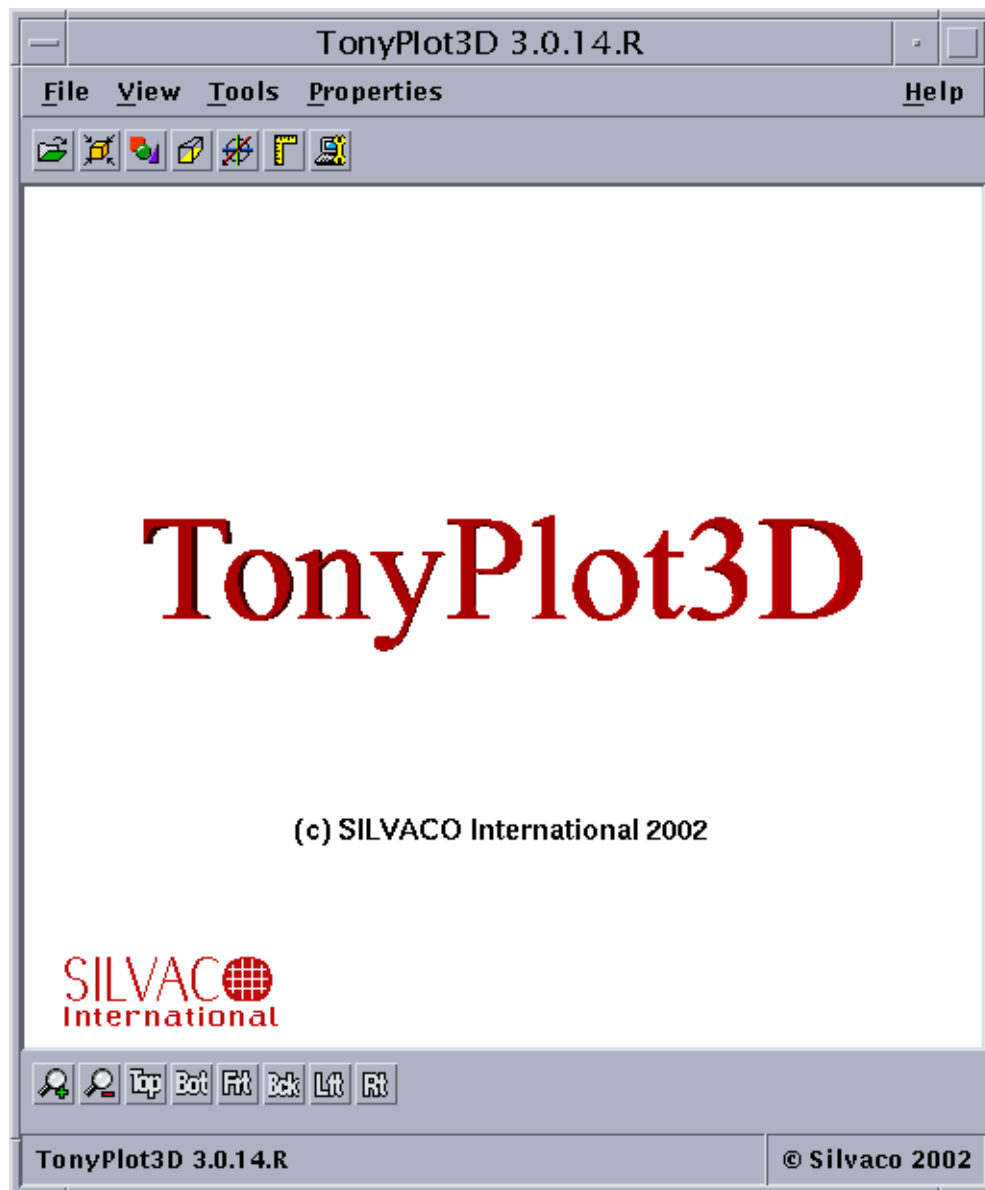


Figure 8-1: TonyPlot3D Title Screen

8.2: Differences Between TonyPlot3D and TonyPlot

TONYPLOT3D only loads and displays 3D data. It doesn't, however, recognize or display 2D data. TONYPLOT, consequently, doesn't recognize 3D files. Sometimes, TONYPLOT may load 3D files and display a cross section of the data. This behavior, however, isn't supported and shouldn't be expected for all 3D files.

While TONYPLOT is capable of loading and displaying several data sets simultaneously, TONYPLOT3D can only show one data set, which means you can only load one file at a time. Whenever a new file is loaded, the old plot is replaced with the new one.

8.2.1: Starting TonyPlot3D

To start TONYPLOT3D, type:

```
tonyplot3d
```

As with TONYPLOT, the data file name that is being loaded can be supplied immediately.

Note: The filename shouldn't begin with a minus (–) character.

Table 8-1 shows a set of command line arguments that can be specified.

Table 8-1: TonyPlot3D Command Line Arguments	
Command Line Option	Description
-buffer [single double]	Specifies the buffering mode to use when drawing the 3D Scene (i.e., single or double).
-help	Display the command line options in the standard output.
-nohw	Forces TONYPLOT3D to avoid using any hardware graphics accelerators that may be available on the computer. By default, TONYPLOT3D tries to use any acceleration it can find. More information about various graphics hardware can be found in Section 8.7: "Operating Platforms".

8.3: Main Window

When TONYPLOT3D is started, the Main Window should appear. Initially, the main window displays a TONYPLOT3D banner with copyright information. Clicking on the banner will start spinning the 3D text. Click in the plotting area again to stop the movement. The layout of the Main Window is shown in Figure 8-2.

The Main Window provides a number of drop down menus and shortcut toolbars to access the features provided within TONYPLOT3D. Table 8-2 describes the different areas of the Main Window.

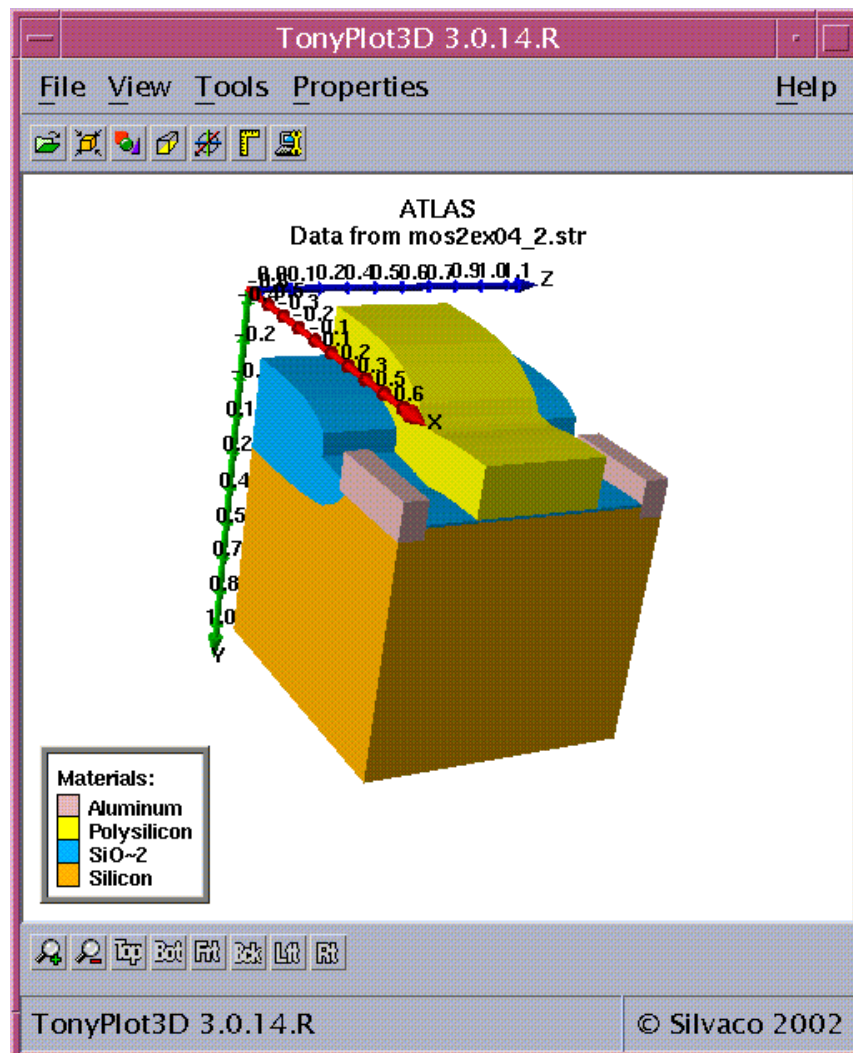


Figure 8-2: Main Window

Table 8-2: Main Window Options

Option	Description
Main Menu	The menus along the top of the TONYPLOT3D window provide access to pop-ups, switches and toggles. These provide detailed control over the display and setup. Some menu options have “hot-keys.” These hot-keys are denoted with an underscore. Use these hot-keys to directly access the features. To use the hot keys, press the ALT key simultaneously with the identified hot key letter (i.e., the underscored letter). Once you do this, press the denoted hot-key letter to access a specific option. For example, to open the D isplay option in the View menu, press: ALT and V , followed by D only.
Main Toolbar	This toolbar complements the pull down menus by providing quicker access to most frequently used options.
Plotting Area	The central portion of the main window is used to display data. In this plot area, the mouse pointer can be used to manipulate the plot.
Plot Control Toolbar	This toolbar provides quick access to controls that change the way the plot area displays the data (i.e., zoom & view transformations).

8.3.1: Main Menu Options

The main menu is described in Table 8-3.

Table 8-3: Main Menu Options








Menu Title	Option	Description
<u>F</u> ile	<u>O</u> pen	A file browser is displayed that allows you to load a 3D structure file.
	Save Surface	Saves all the faces of the 3D elements as triangles in a .str file.
	Save as JPEG	Saves the bitmap of the Plotting Area in a JPEG file format (compression level 75%). The size of the saved bitmap matches the size of the plotting area.
	Save as TIFF	Saves the bitmap of the Plotting Area in a TIFF file format. The size of the saved bitmap matches the size of the plotting area.
	<u>E</u> xit	Exits from Tonyplot3D
<u>V</u> iew	<u>C</u> enter	Centers the structure in the Plotting Area with a pre-defined pose matrix.
	Zoom <u>I</u> n	Zooms into the current structure display.
	Zoom <u>O</u> ut	Zooms out of the current structure display.
	<u>S</u> how From	The Show From option has sub-options. These sub-options are: Top, Bottom, Front, Back, Left & Right . All these sub-options simply move the current display structure to display it from the selected position.

Table 8-3: Main Menu Options

Menu Title	Option	Description
<u>T</u> ools	<u>D</u> isplay	Opens the Display Modes Window. This is used to change the display mode and viewing parameters.
	<u>O</u> bject Editor	Displays a hierarchy of the objects in the current scene and structure.
	<u>C</u> utplane	Starts the Cutplane Tool and shows the 2D cut as it would be exported to TONYPLOT.
	<u>P</u> robe	Starts the Probe Tool and shows the picked object using the left mouse button.
<u>P</u> roperties	<u>R</u> uler	Starts the Ruler Tool and shows the quantities of the picked object using the Control key and the left mouse button.
	<u>C</u> amera	Edits camera settings.
	<u>C</u> olors	Edits color settings of the structure and the plotting area.
	<u>F</u> onts	Edits font settings of the plotting area.
	<u>L</u> egend	Edits the legend settings.
	<u>L</u> ights	Edits lighting settings.
	<u>M</u> ouse	Edits mouse and automatic movement settings of the plotting area.
	<u>S</u> tructs	Edits outline, mesh, transparency and drag type settings of the structure.
	<u>L</u> oad	Loads the local set of properties in \$HOME/.masterrc file.
	<u>S</u> ave	Saves the set of properties locally in \$HOME/.masterrc file.
	<u>R</u> eset Default	Resets the properties to the default from the installation directory.
<u>H</u> elp	<u>A</u> bout Tonyplot3D	Shows TONYPLOT3D version information.
	TonyPlot3D Help	Displays the user's manual for TONYPLOT3D (PDF format).
	<u>R</u> elease Notes	Displays the release notes for TONYPLOT3D (PDF format).









8.3.2: Main Toolbar Options

Table 8-4 shows the main toolbar options.

Table 8-4: Main Toolbar Options		
Icon	Name	Effect
	File Open	A file browser is displayed that allows you to load a 3D structure file.
	Center	Centers the structure in the Plotting Area with a pre-defined pose matrix.
	Object Editor	Displays a hierarchy of the objects in the current scene and structure. Once this hierarchy is displayed, you can then perform actions such as Hide or Show. For more information about the Object Editor, see the Section 8.5.1: "Object Editor".
	Cutplane	Starts the Cutplane Tool and shows the 2D cut as it would be exported to TONYPLOT.
	Probe	Starts the Probe Tool and shows the picked object using the left mouse button.
	Ruler	Starts the Ruler Tool and shows the quantities of the picked object using the Control key and the left mouse button.
	Configure Display	Opens the Display Modes Window. This is used to change the display mode and viewing parameters.

8.3.3: Plot Control Toolbar Options

Plot control toolbar options are described in Table 8-5.

Table 8-5: Plot Control Toolbar Options		
Icon	Tooltip	Effect
	Zoom In	Zoom into the structure.
	Zoom Out	Zoom out from the structure.
	Show Top	Show the structure from the top.
	Show Bottom	Show the structure from the bottom.
	Show Front	Show the structure from the front.
	Show Back	Show the structure from the back.
	Show Left	Show the structure from the left.
	Show Right	Show the structure from the right.

8.3.4: Plot Control Using the Mouse

Within the Main Window, you can use the mouse to move the structure. You can adjust the functions of the mouse by using the Mouse Interface (see "Mouse" section on page 8-38 for more information).

There are six different movements you can apply to the structure: rotation (which has three modes), translation, zoom, and scaling.

Rotation

The left-button rotates the structure at the center of its bounding box. Shift and the left-button rotates from the point where the click originated in the 3D scene. Shift and the middle-button rotates the structure from an axis going out of the screen and from the middle of the viewport.

Translating

To translate (move) the plot, hold down the middle-button while dragging the mouse. The plot will then move in response to the movements of the mouse.

Zoom

To zoom, use the right-button. This is essentially a translation applied on an axis going out of the screen. Very different from the scaling, since the vertices of the structures aren't modified.

Scaling

To scale a plot, hold down the Shift key and use the right-button. The scale can be applied independently in the X, Y and Z directions (for more information, see the **Scaling** option in the Camera Interface on page 34). A scaling is applied to the vertices. This movement is useful when one dimension of the structure has to be scale to see its details. This happens when viewing very thin structures.

Automatic Movements

You can rotate, translate, zoom, and scale automatically by releasing the mouse button just before stopping the mouse for any of the above mentioned operations. To enable automatic movements, check the **Automatic Movements** box in the Mouse Interface. See "Mouse" Section on page 38 for more information.

Note: The bounding box will only appear if you activate it in the Struct Interface. See "Structure" section on page 8-39 for more information.

8.4: Display Modes

You can view or configure the display of the plot by using the Display Modes Window (see Figure 8-3). To open the Display Modes Window, select **View→Display**. In this window, there are five different display modes: **Regions**, **Contours**, **Rays**, **Isosurfaces**, and **Vectors**. These display modes are all non-exclusive. To activate a display mode, simply turn on its toggle button.

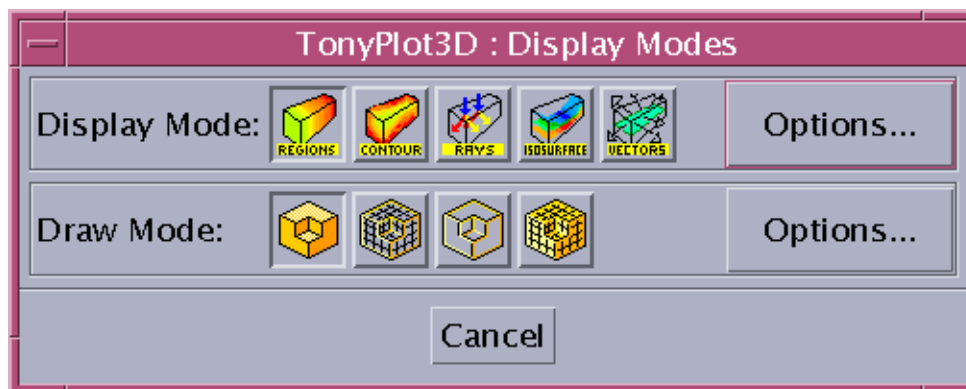






Figure 8-3: Display Modes Window

To further configure the appearance of the plot, choose the **Options** button in the upper section of the Display Modes Window. The Display Mode Options Dialog will appear with additional settings for each mode (see the next subsections for more information on these settings). Choose the tabs relevant to the mode you want to configure. Don't forget to click on the **OK/Apply** button to apply your changes. When using these settings, make sure you turn on the relevant display mode.

The Draw Mode (shown in the bottom half of the Display Modes Windows) controls the way the faces of a structure are displayed. Table 8-6 describes these modes.

Table 8-6: Draw Modes		
Icon	Mode	Description
	Solid Fill	The solid fill mode shows all the exterior faces of the object.
	Edges Only	In this mode, only the exterior sharp edges of the object are shown.
	Mesh	This mode shows the mesh of the external faces by default. You can change this mode to display all the mesh in the structure or in a volume specified by a cylinder. The possible settings are: Faces , Elements (tetrahedra or prisms), or Volume (Cylinder). See Figure 8-4.
	Solid Fill With Mesh	This is a combination of the solid fill mode and the mesh mode (i.e., it fills all the cells that make up the object).

To further configure the appearance of the mesh of the structure, choose the **Options** button in the lower section of the Display Modes Window. The Draw Mode Options Dialog shown in Table 8-4 will appear.

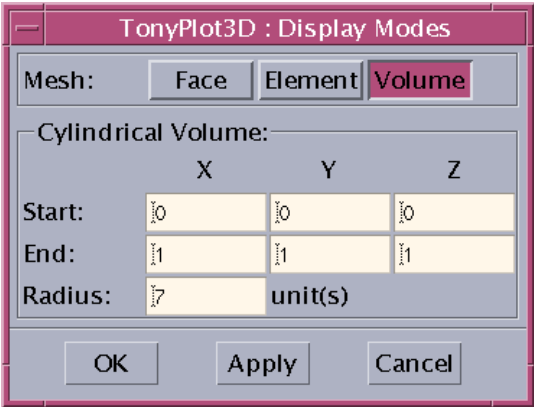


Figure 8-4: Draw Mode Options Dialog Box

Table 8-7 shows the different options available in the Draw Mode Options Dialog.

Table 8-7: Draw Mode Options Dialog Box	
Option	Description
Face	Draws the mesh for the surface of the regions only.
Element	Draws the mesh for all the elements (3D).
Volume	Draws the mesh in the specified cylindrical volume. Use the X,Y,Z , and Radius fields to specify the cylindrical volume.

8.4.1: Regions

The Region Display Mode (Figure 8-5) is the default display mode. If you select **Material**, each material will be assigned a color. If you select **Region**, one color will be assigned to each region in the data. In either case, a legend is provided to show each color assigned to material or region names.

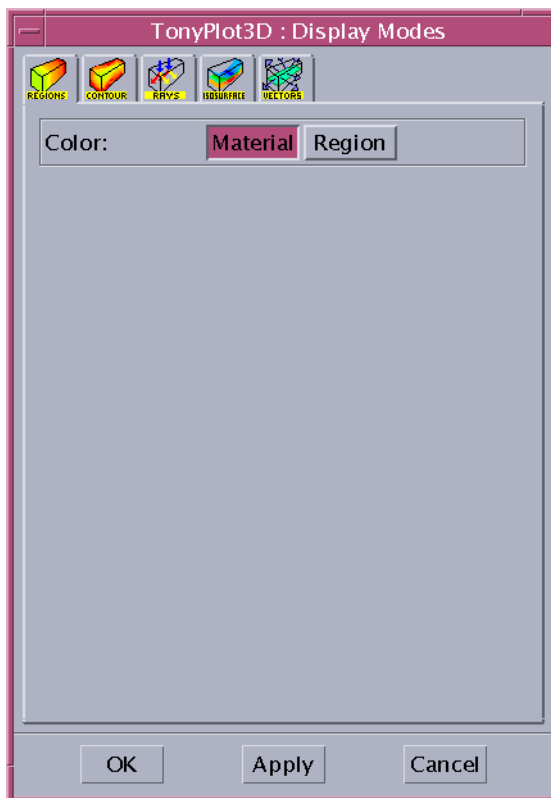


Figure 8-5: Regions Display Mode

8.4.2: Contours

The Contour Display Mode is shown in Figure 8-6. Contours are shown as a colorization of the exterior faces of an object. Contours are drawn only on exterior portions of the structure. Figure 8-7 shows an example of these contours. To observe the inside values perform a cutplane (see Section 8.5.2: “Cutplane” for more information).

The **Quantity** option box holds all of the quantities present in the data set. Choose one of these quantities for contouring.

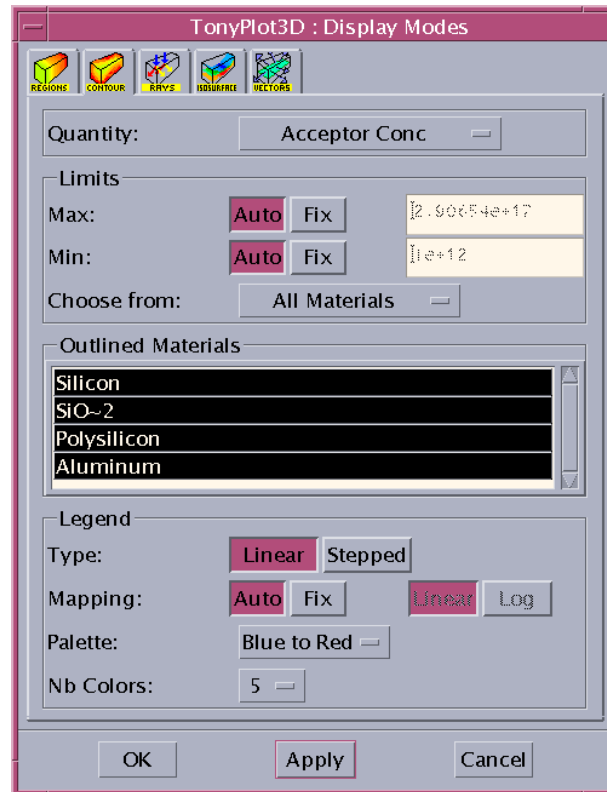


Figure 8-6: Contours Display Mode

Data Constraint Settings

The **Limits** Group Box controls the data range used in the contours legend. If you set **Min** and **Max** to **Auto**, then TONYPLOT3D extracts the data range from the different materials of the structure. You can use the pulldown menu **Choose From** to change the materials used in the computation of this data range. The three choices are: **All Materials**, **Outlined Materials** and **Selected Objects**. **All Materials** is the default settings. In this mode, all the nodes in the structure are used to find out the data range. If **Outline Materials** is selected, then the list of highlighted materials in the **Outlined Materials Group Box** are chosen. If **Selected Objects** is selected, the data range is computed from the list of selected objects in the scene (see Section 8.5.1: “Object Editor”). The limits can also be user-defined by turning on the **Fix** buttons and specifying the **Min** and **Max**.

The **Legend** Group Box controls the way the contours are drawn on the faces. If you use **Linear Type**, the colors will linearly interpolate between the reference colors of the legend. The **Stepped Type** will produce a finite number of colors to use within the data range. This mode is very useful for color-blind users. The **Palette** and **Nb Colors** are also available to further change the appearance of the legend.

The quantity values can either be plotted with a linear or logarithmic scale. When the mapping is linear, the values are directly mapped to the legend's colors. When the mapping is logarithmic, the \log_{10} is taken before the mapping occurs. Figure 8-6 shows an example of logarithmic mapping for the Donor Concentration. The default mapping is chosen by TONYPLOT3D but can be overridden by using the **Fix** button.

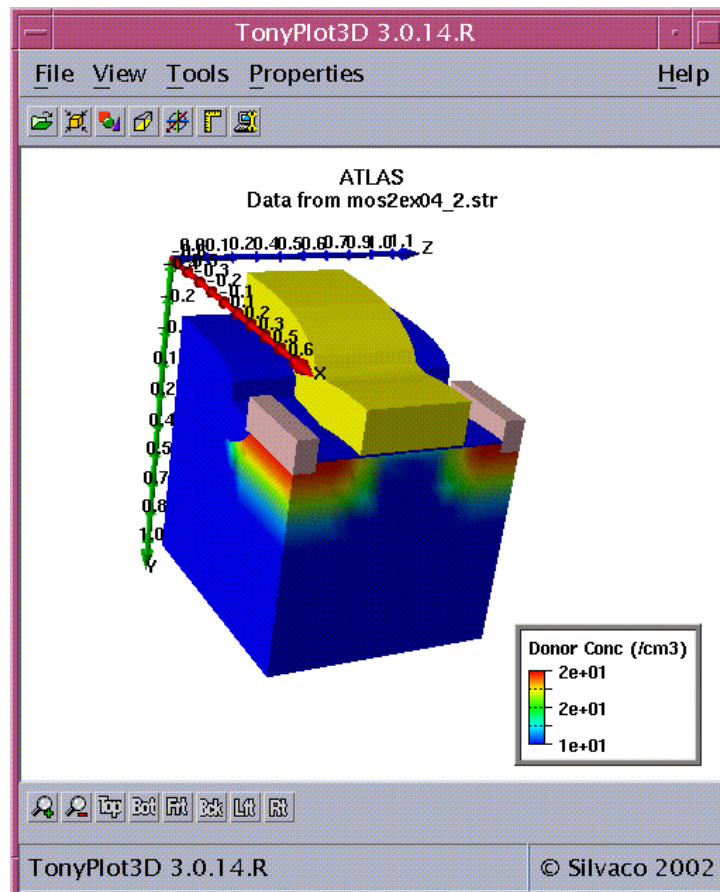


Figure 8-7: Contours in a plot

8.4.3: Rays

The Ray Display Mode shows a list of all the rays contained in a structure file. When highlighted in the list and after pressing the **Apply** button, the rays are going are drawn in the Plot Area. The **Ray Settings** Group Box controls the way the rays are displayed. The rays can be displayed as lines or as cylinders. When drawn as lines, use the Line Width Pull-Down menu to change the thickness of the rays. When drawn as cylinders, use the Cylinder Radius Slider to change the relative size of the cylinder's radius. Use the CTRL key to select multiple rays in the list.

Figure 8-9 shows an example of rays displayed as lines.

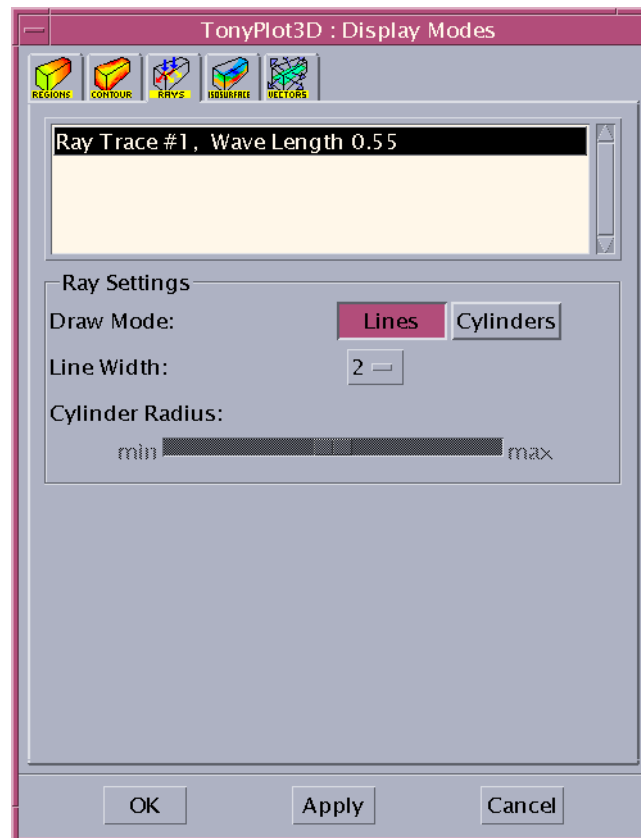


Figure 8-8: Ray Trace Display Mode

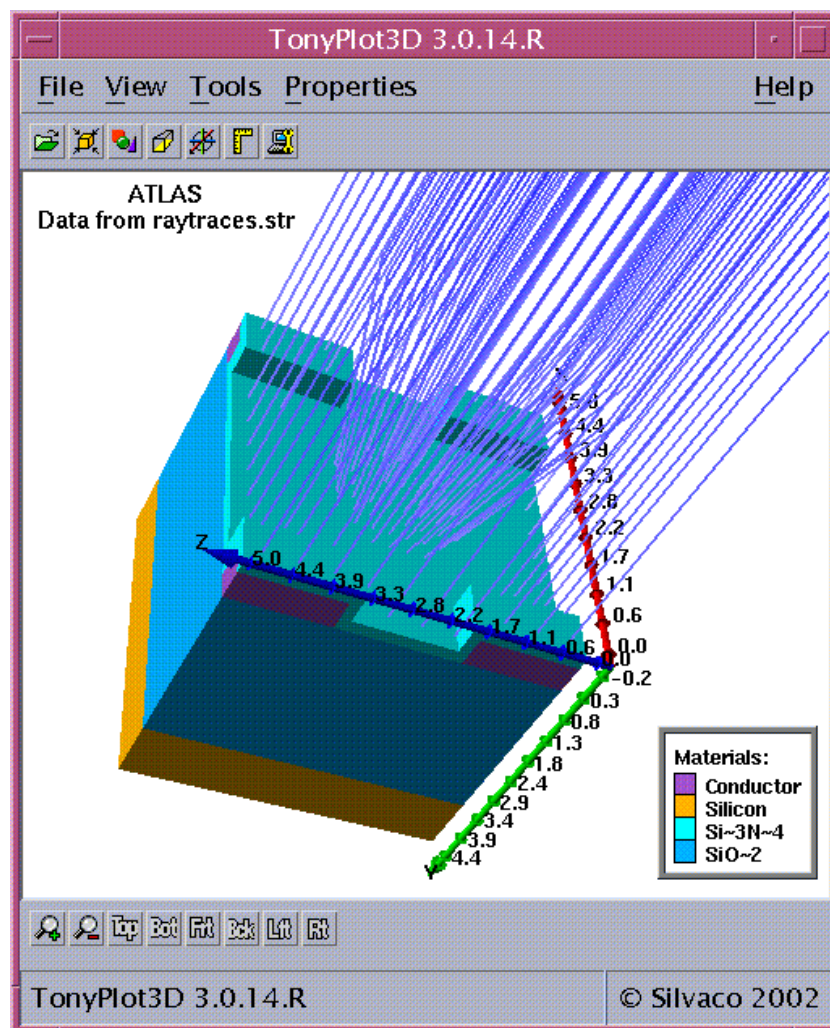


Figure 8-9: Example of rays in a plot

8.4.4: Isosurface

The Isosurface Display Mode is used to show surfaces of constant value throughout a 3D structure. The **Quantity** controls which impurity is used in the computation. The value of an isosurface has to be within the data range (**Min,Max**) of the **Quantity** and can be changed in the **Value** text field.

To view an isosurface, you can either turn on the **Preview IsoSurface** or select the **Create** button. Once created, the isosurface will appear in the isosurface list. Use the **Delete** button to remove the selected isosurface from the list. You can use the Draw Mode to change the appearance of all the isosurface at once (see Table 8-6).

Figure 8-10 shows a MOSFET that has been hit with SEU (Single Event Upset) alpha particle strikes. The path of the particle is shown by the isosurface plots of electron concentration.

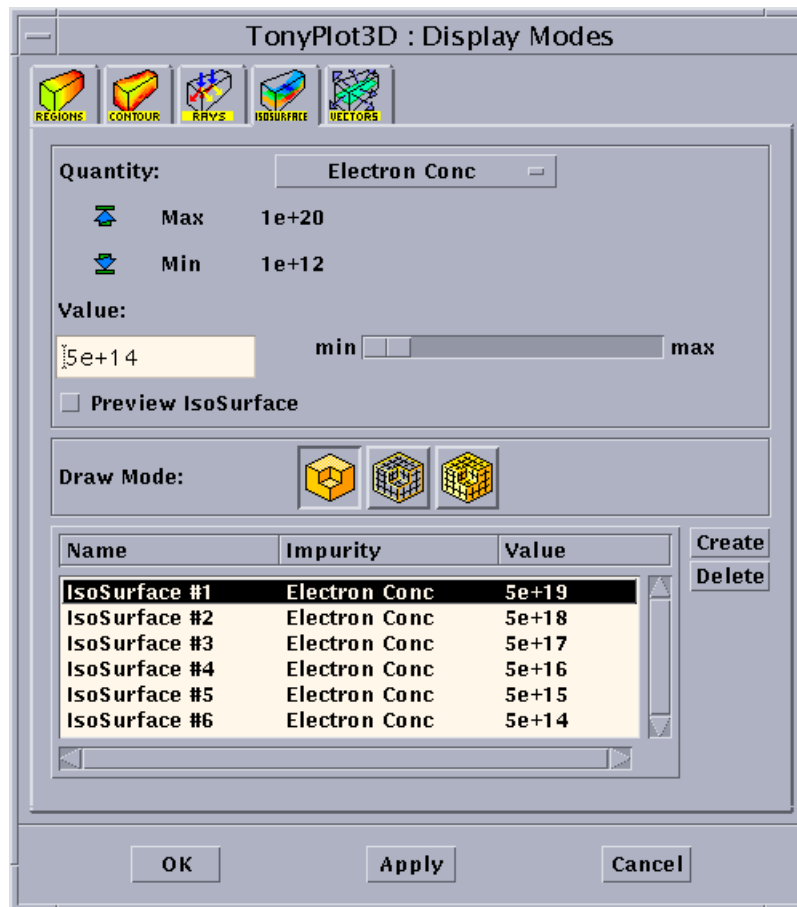


Figure 8-10: Isosurface Display Mode

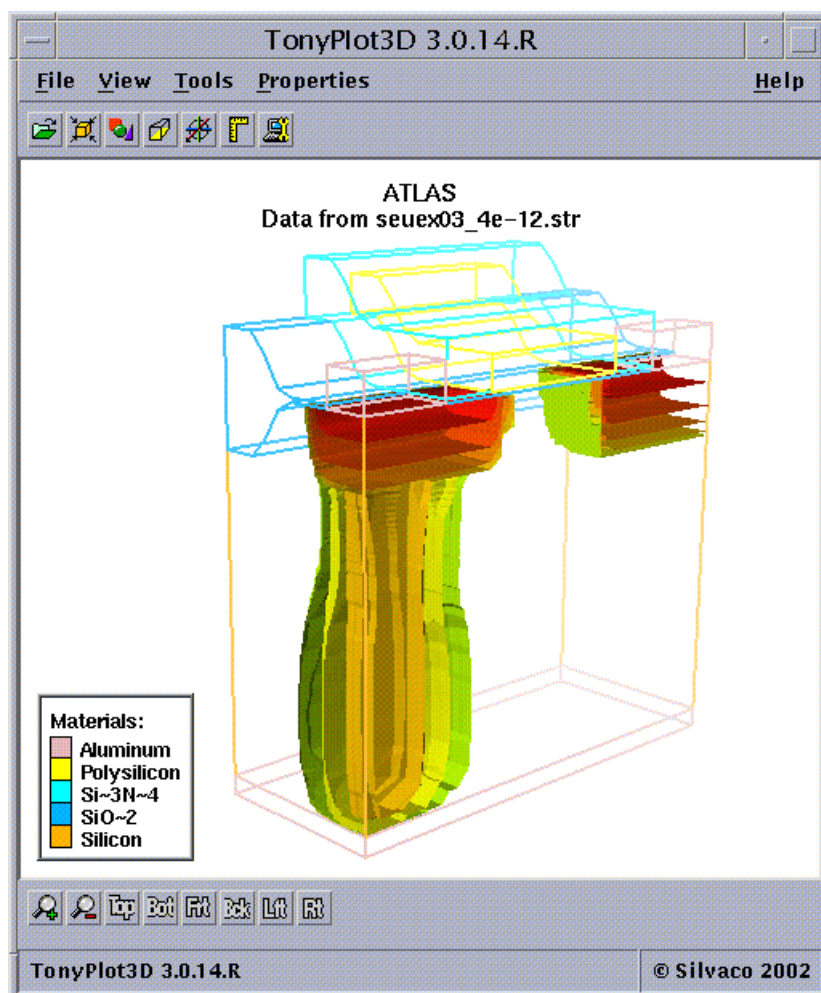


Figure 8-11: Example of Isosurfaces in a plot

8.4.5: Vectors

When a structure contains vector data, you can use this method to visualize their directions and magnitudes. Figure 8-12 shows the Vectors Display Mode.

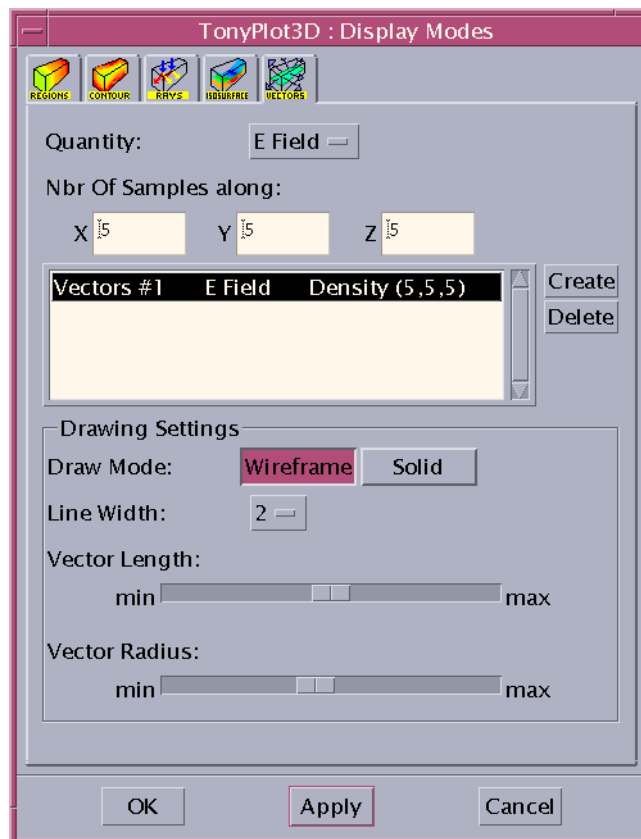


Figure 8-12: Vectors Display Mode

Use the **Quantity** menu to display a particular vector quantity. The number of samples along each axis can be altered with the respective box for x, y, and z.

Once you choose the particular vector to be displayed, it has to be created. To create it, click on the **Create** button. The dialog box describing the vector quantity will appear next to the **Create** button (see Figure 8-12). To have several vector quantities displayed simultaneously, create each vector separately.

You can control the way vectors are displayed by using the Draw Mode. In wireframe mode, a line is used to display the vectors. In Solid Mode, the vectors are drawn as cylinders and cones (solid arrows).

Linewidth changes the thickness of the vector lines. **Vector Length** and **Vector Radius** change the length and radius of the vectors respectively. The radius corresponds to the size of the arrow at the end of the vector.

Figure 8-13 shows an example of vectors in a plot.

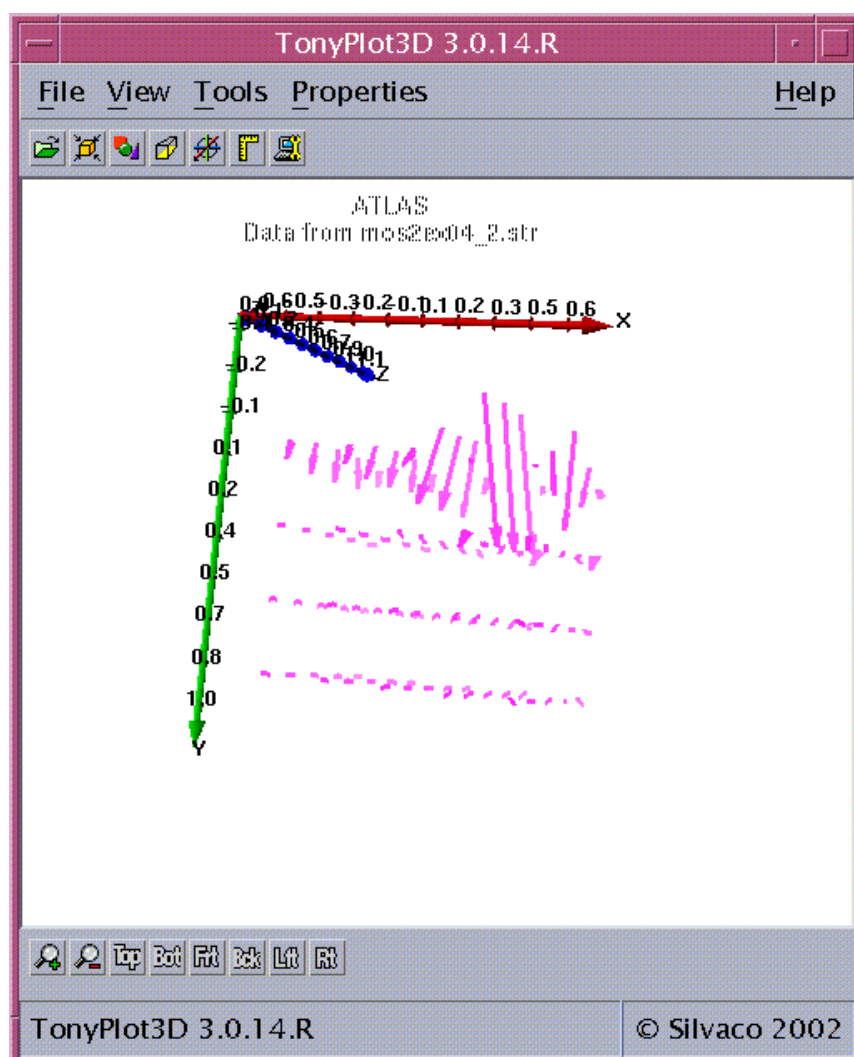


Figure 8-13: Example of Vectors in a plot

8.5: Tools

8.5.1: Object Editor

The Object Editor displays the objects in the current scene. The components of the structure follow a hierarchal tree formation (see Figure 8-14). Depending on what data is present in your structure and what is viewed, some of the objects listed below may not appear in the tree.

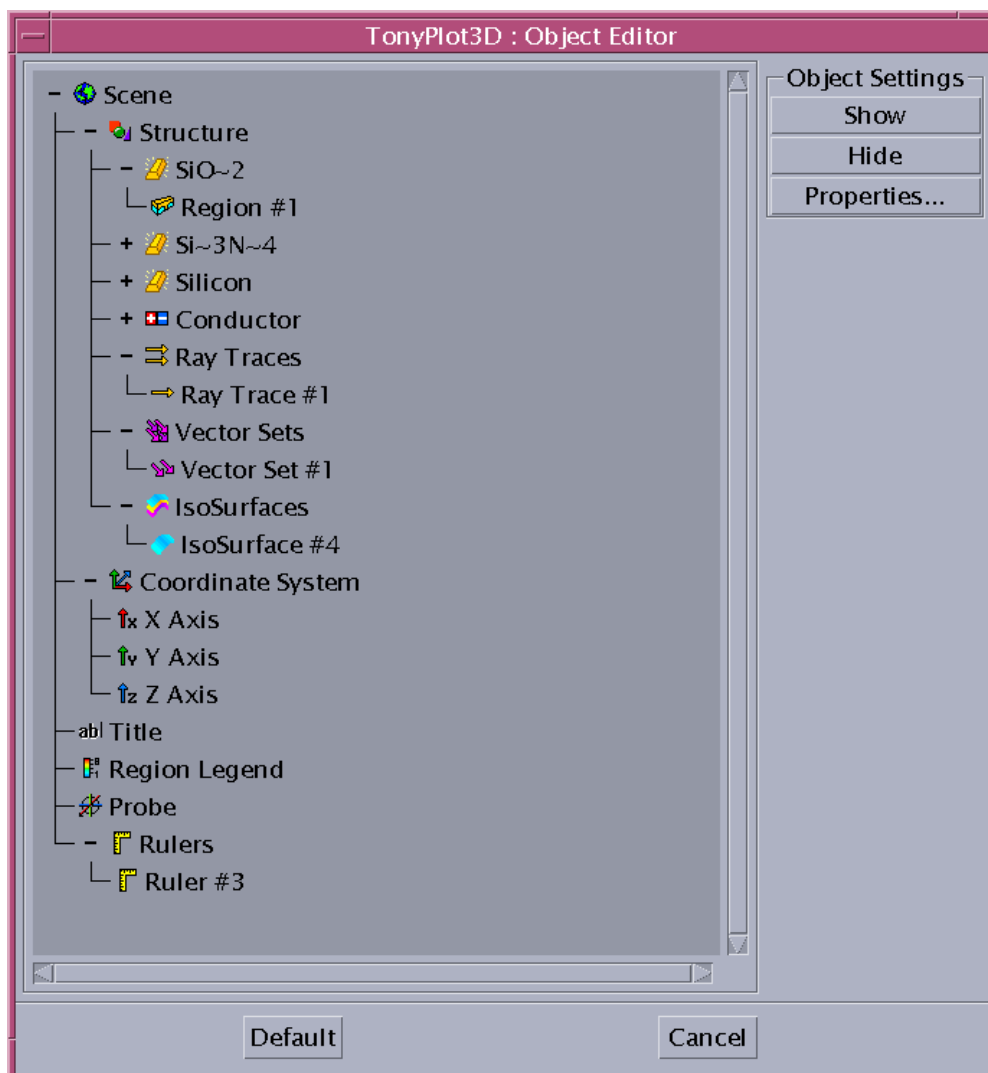


Figure 8-14: Object Editor

Table 8-8 lists and describes each part of the hierarchy.


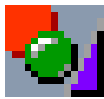



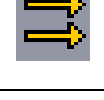
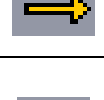



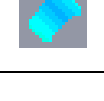
Table 8-8: Object Hierarchy Structure		
Icon	Type	Description
	Scene	This is the top level object within TONYPLOT3D's display. This is everything that appears in the Main Window. Note: Only one scene is currently supported.
	Structure	This is the main structure object. It is the parent object that contains all of the materials and regions. Note: Currently, only one structure per scene is supported
	Material	Each structure is broken up into a number of different materials (e.g., silicon, polysilicon, and so on).
	Region	Each material is then broken into a number of distinct regions. This is the lowest level object displayed in the object dialog.
	Electrode	Identifies the electrode settings.
	Ray Traces	Displays the ray trace settings.
	Ray Trace #	Displays the setting of an individual ray trace.
	Vector Sets	Displays the properties of the vector sets.
	Vector Set #	Displays the individual vector set.
	Isosurfaces	Displays the properties of the isosurfaces.
	Isosurface #	Displays the properties of the individual isosurface.







Table 8-8: Object Hierarchy Structure		
Icon	Type	Description
	Coordinate System	Identifies the coordinate systems axis.
	Title	Displays the Title properties for viewing and modifying.
	Legend	Displays the Legend properties for viewing and modifying. Note: Actually, there are two legends. One for the contours (Contours Legend) and the other for the regions (Region Legend).
	Probe	Displays the Probe properties
	Rulers	Displays the Ruler properties.
	Ruler #	Displays the properties of an individual ruler.

Table 8-9 shows the mouse actions that can be used in the object tree.




Table 8-9: Mouse Action Functionality	
Mouse Action	Description of Functionality
	The left mouse button is used on an icon to expand/shrink that branch of the tree. You can also use this button over an object name to select it, while deselecting all others.
	The middle mouse button is used to select/deselect an object name in the tree. Several objects can be selected at once by clicking on them. You can also use this button over an icon to expand/shrink that branch of the tree.
	The right mouse button is used on an object to display a menu, which allows you to change certain object settings. See Table 8-10 for a description of these menus.

Table 8-10: Right Mouse Menus

Object	Description	Menu Option	Description
Scene	Applies to itself and all its children.	Show	Displays the entire scene.
		Hide	Omits the entire scene.
		Properties	Opens the Display View Mode (see Section 8.4: “Display Modes” section).
Structure	Applies to the structure and all its children. The right mouse menu options, however, are the same for the structure and its children (apart from ray traces, vector sets and iso-surfaces, these are detailed separately). Therefore, the word “structure” is interchangeable with “children”.	Show	Displays the entire structure.
		Hide	Hides the structure.
		Opaque	Causes the structure to be opaque.
		Transparent	Causes the structure to be transparent.
		Solid	Causes the structure to be solid.
		Meshed	Displays the mesh for the structure.
		Edges	Displays only the edges of the structure.
		Solid and Meshed	Causes the structure to appear solid and meshed.
		Properties	Opens the Object Properties Pop-up, where you can modify all the above menu options.
Structure: Ray Traces	Displays the rays’ properties.	Show	Displays the rays.
		Hide	Hides the rays.
		Properties	Allows you to show or hide the rays. You can also change the draw mode (line or cylinder) and the linewidth.

Table 8-10: Right Mouse Menus

Structure: Ray Trace #	Displays an individual ray's properties.	Show	Displays the ray.
		Hide	Hides the ray.
		Properties	Allows you to show, hide, or change the color of a ray.
Structure: Vector Sets	Displays the vector sets' properties.	Show	Shows the vector sets.
		Hide	Hides the vector sets.
		Properties	Allows you to to show or hide the vector sets. It also allows you to change the draw mode (wireframe or solid) and linewidth.
Structure: Vector Set#	Displays a vector set's properties.	Show	Shows the vector set.
		Hide	Hides the vector set.
		Properties	Allows you to show, hide, or change the color of a vector.
Structure: Isosurfaces	Displays isosurfaces' properties.	Show	Shows the isosurfaces.
		Hide	Hides the isosurfaces.
		Properties	Allows you to show or hide the isosurface.
Structure: Isosurface#	Displays an individual isosurface's properties.	Show	Shows the isosurface.
		Hide	Hides the isosurface.
		Properties	Allows you to show, hide, or change the color of an isosurface.
Coordinate System	Identifies the coordinate systems axes. The axes (X,Y, and Z) also have their own menus. These menus perform similar functions, except the functions pertain to the individual axis rather than the entire axes of the system.	Show	Shows the axes.
		Hide	Hides the axes.
		Properties	Allows you to show or hide the axes and change the numerical precision of them. For the individual axis (X,Y or Z), you can show, hide, and change the color of the axis and its label. You can also show, hide, and change the number of tick (increment) marks in the axis.

Table 8-10: Right Mouse Menus

Title	Displays the Titles' properties.	Show	Shows the Title.
		Hide	Hides the Title.
		Properties	<p>Allows you to show, hide, move, and change the color of the title. You can also move the title by hovering the mouse pointer over the title and pressing Shift. Use the left mouse button and drag it to the desired location.</p> <p>Also in this menu, you can change the name of the title and the subtitle.</p> <p>The position of the title can either be constrained or floating. If it's constrained, it remains at its position when you resize the screen. If it's floating, the position changes with the size of the screen.</p>
Legend	Displays the Legend's (Contours and Region) properties.	Show	Shows the Legend.
		Hide	Hides the Legend.
		Properties	<p>Allows you to show, hide, move, and change the color of the legend. You can move the legend in the same way as the Title, which is described above.</p> <p>The position of the Legend can either be constrained or floating, which does the same thing as for the Title.</p>
Probe	Displays the Probe's settings.	Show	Shows the Probe.
		Hide	Hides the Probe.
		Properties	Allows you to show or hide the Probe. You can also change the color of the probe and its axes.
Ruler	Displays the Rulers' settings.	Show	Shows the rulers.
		Hide	Hides the rulers.
		Properties	Allows you to show, hide, and change the precision of the rulers.

Table 8-10: Right Mouse Menus

Ruler#	Displays the settings of an individual ruler.	Show	Shows an individual ruler.
		Hide	Hides an individual ruler.
		Properties	Allows you to show, hide, and change the color of an individual ruler. You can also show, hide, and change the color of the individual ticks (increment marks) and their labels.

8.5.2: Cutplane

The cutplane is a plane (2D slice) that's drawn through a 3D structure. The cutplane may be used so you can examine the inside of a structure or to perform 2D device within ATLAS. An example of a cutplane is shown in Figure 8-15. For more information about ATLAS, see the ATLAS USER'S MANUAL.

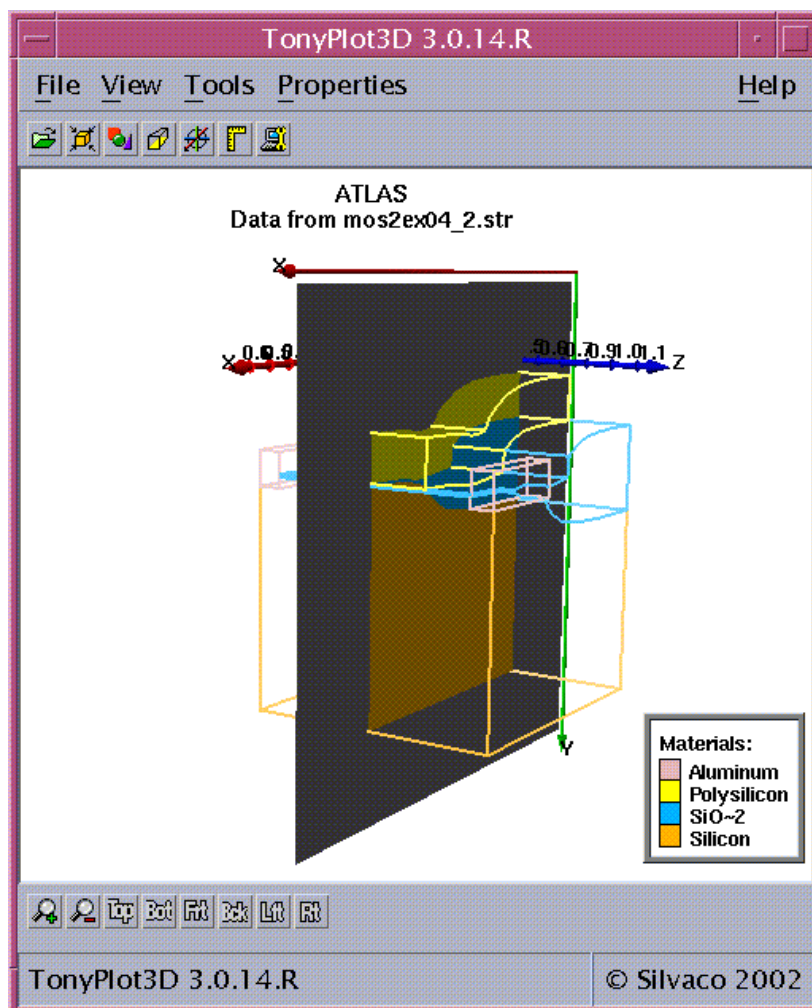
**Figure 8-15: An example of a cutplane**

Figure 8-16 shows the Cutplane Dialog. The top half of this figure shows the various settings to adjust the cutplane, while the bottom half shows the extracted cutplane.

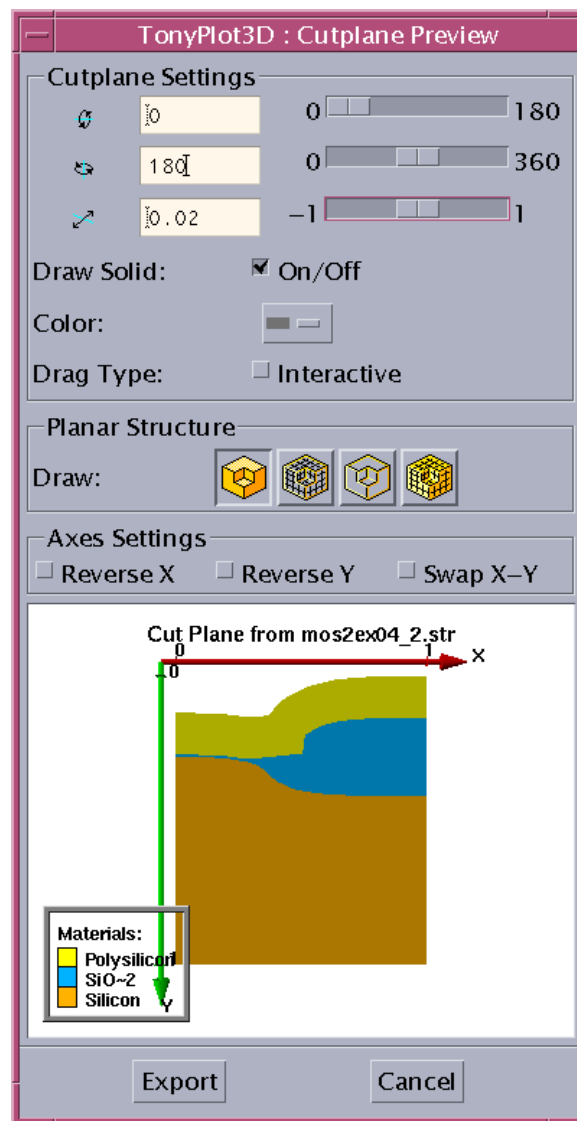


Figure 8-16: Cutplane Dialog

The exact position of the cutplane is chosen through three cutplane settings. The first two settings position the angle of the plane relative to the axis. The third setting sets the spatial position relative to the normal on the cutplane.

The cutplane itself can be drawn solid to visually aid its positioning. You can also alter the color of the cutplane.

While positioning the cutplane, you can monitor the positioning continuously as it changes or update it after it has been moved. To do this, toggle the **Drag Type** switch (i.e., **Interactive** or not respectively).

You can also alter the display of the cutplane by using any of the draw modes previously described in Table 8-6.

Axes Settings allows you to reverse the x axis, the y axis, or swap them over.

Once you've obtained the desired plane, you can export it directly to a file or TONYPLOT by using the Export Slice Dialog. Press the **Export** button to open the Export Slice Dialog (Figure 8-17). If it's going to be exported to a file, enter the file name in the appropriate box.



Figure 8-17: Export Slice Dialog

One or several planes can be exported at once, depending on the **Export Slice** switch. To export several planes, adjust the **Min** and **Max** settings. These settings correspond to the start and end of the spatial position of the cutplanes.

The **Slice Settings** can be set to **Auto** or **Step**. If it's set to **Auto**, you need to choose the number of steps. If it's set to **Step**, then you need to define the step size.

Figure 8-18 shows five cutplanes exported to TONYPLOT with the **Min** set to -1 and the **Max** set to 1, respectively throughout the structure.

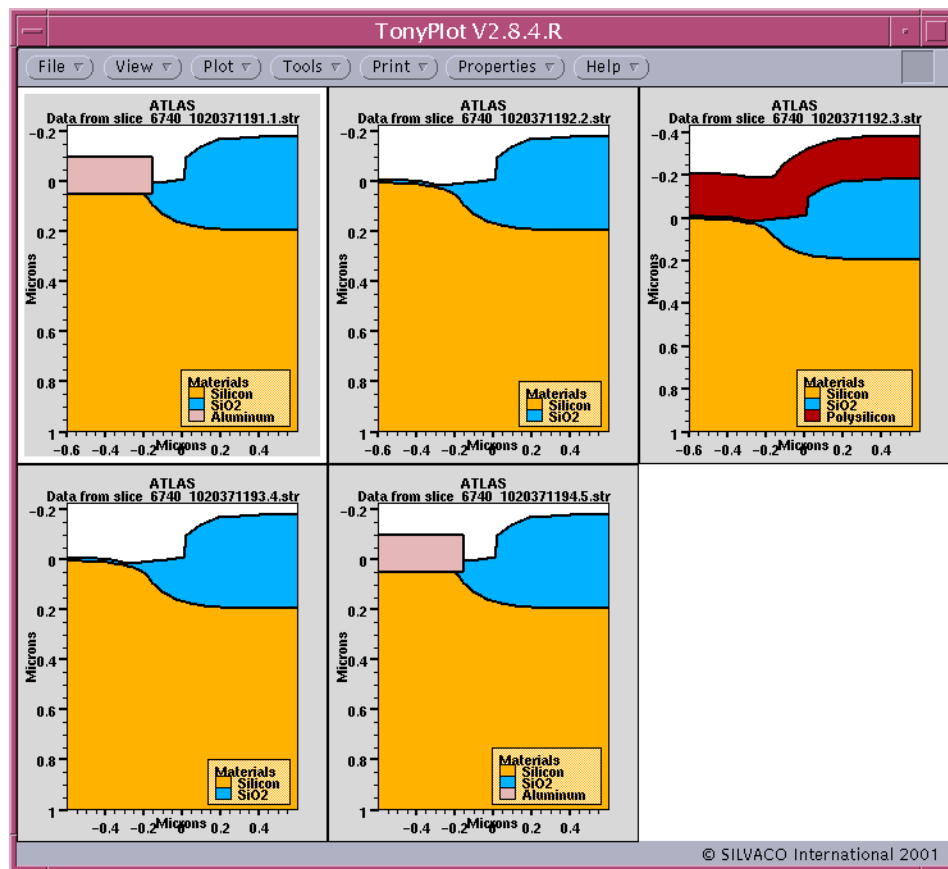


Figure 8-18: An example of five exported cutplanes

8.5.3: Probe

The Probe Tool allows you to probe any point within the structure. To probe a point, click the **Probe** button in the Toolbar.

A particular point in the structure needs to be selected for the probe to extract data for all the quantities present in the structure. When you click on the structure, a sphere appears at the selected location on the structure (see Figure 8-19). The Probe View is then updated with the relevant information (see Figure 8-20).

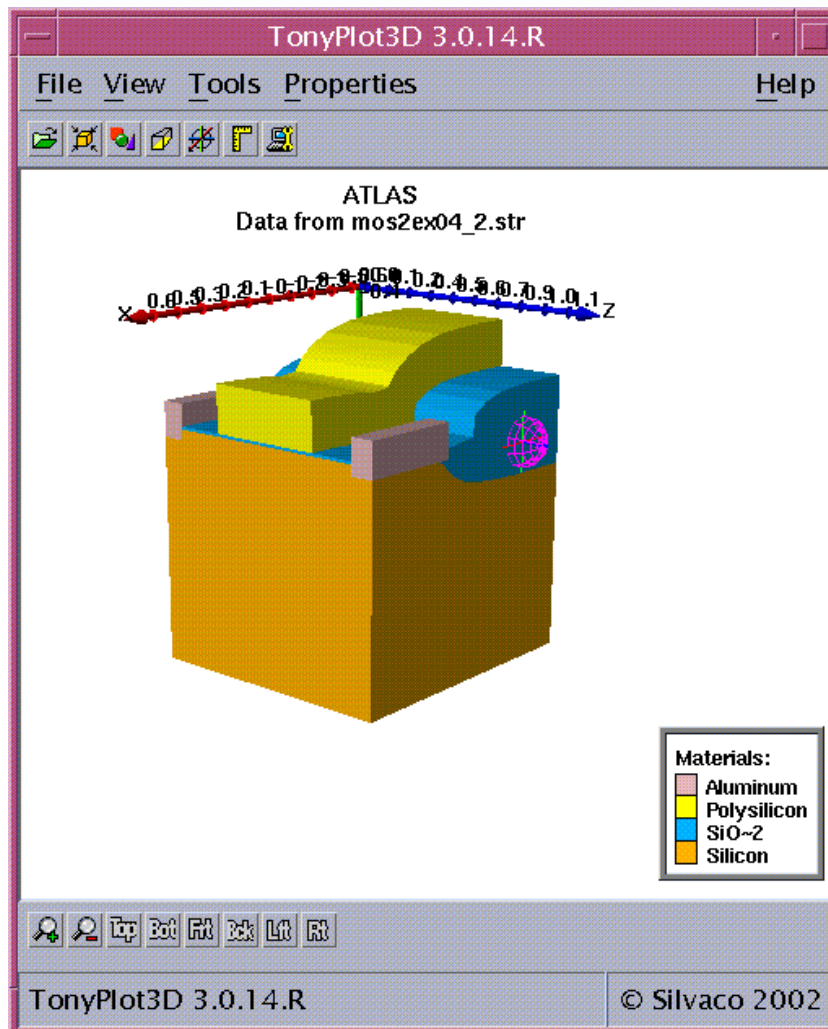


Figure 8-19: An example of a Probed Structure

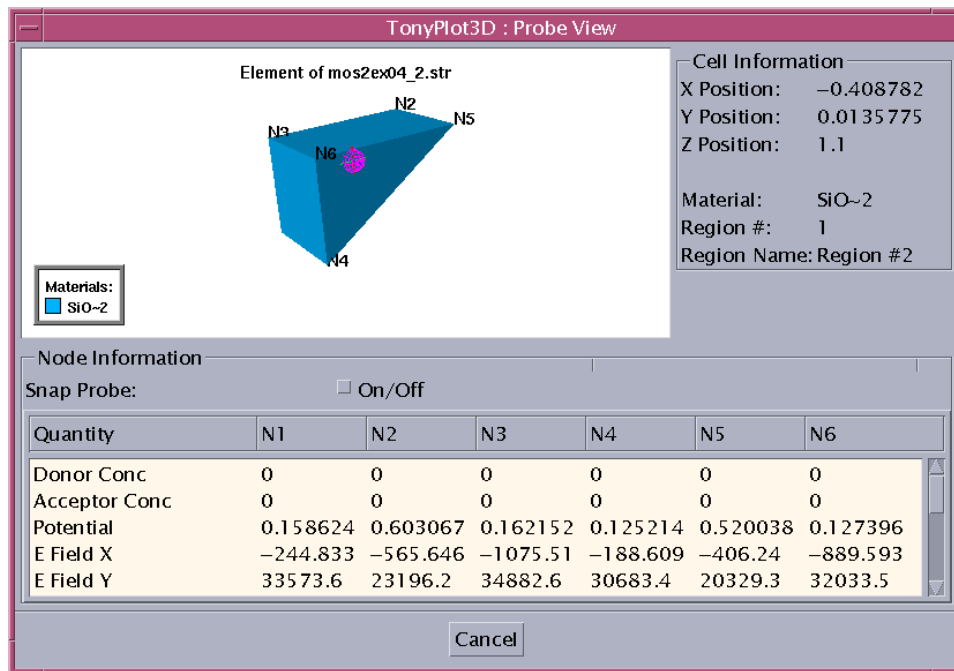


Figure 8-20: Probe View

The sphere and the picked element (prism or tetrahedron) will be drawn with identifiers at its nodes in the Probe View. For each node, quantity values are displayed. The exact position of the probe is also shown with its material and region identification (x, y, and z positions).

You can also use the mouse to move the picked element in the Probe View as described in the "Plot Control Using the Mouse" section on page 8-7.

Note: You can also probe the triangles of the cutplane. The picked triangle is then drawn in the Probe View. The element/face is positioned with the same orientation in the Probe View as it appears in the Main Window.

8.5.4: Ruler

The Ruler Dialog (Figure 8-21) is used to obtain information about quantities within the structure. To use the ruler, hold down the Control key and the left mouse button where you wish to start. Then, drag the mouse and release the button where you wish to stop.

TonyPlot3D : Ruler			
Properties:			
Name:	Ruler #3		
	X	Y	Z
Start:	-0.40566	-0.02134	1.1
End:	-0.22625	0.459388	1.1
Delta:	0.179408	0.480734	0
Length:	0.51312 unit(s)		
Quantities:			
Quantity:	Donor Conc		
Start:	0		
End:	100045		
Delta:	100045		
Rulers:			
Anchor Ruler:	<input checked="" type="checkbox"/> On/Off		
Snap Ruler:	<input type="checkbox"/> On/Off		
Tick Step:			
	Ruler #3		Delete
			Delete All
Cancel			

Figure 8-21: Ruler Dialog

The **Start** XYZ and **End** XYZ positions with the difference between each respective pair, **Delta**, is shown in the top of the dialog (see Figure 8-21). The length of the line, **Length**, is also shown.

Any quantity value present in the structure can also be displayed for the **Start** and **End** points along with their difference in the Ruler Dialog.

To choose a quantity, select it from the **Quantity** menu. You can also make the ruler permanent by toggling the **Anchor Ruler ON/OFF** switch. To anchor a ruler, check the **Anchor Ruler** box before you draw the ruler.

Figure 8-22 shows the ruler drawn from the data in Figure 8-21.

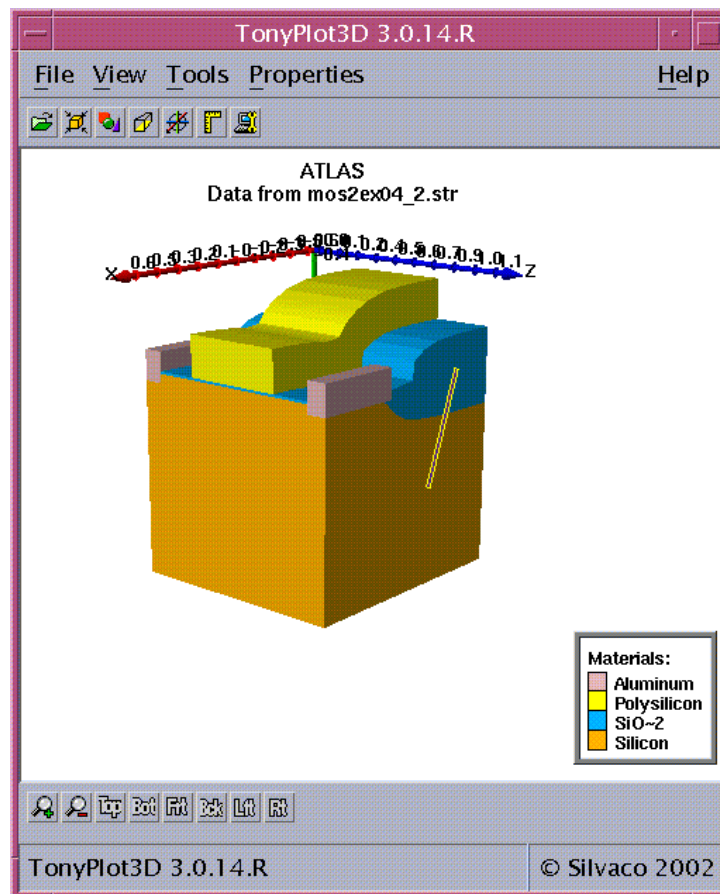


Figure 8-22: An example of a drawn ruler

A ruler identifier will be created. You can anchor several rulers in a structure, each of which has an identifier. You can delete each ruler separately or all at once.

Use the **Tick Step** box if you want to change the number of ticks (increment marks) on the ruler.

You can snap the ruler to existing vertices in the structure by using the **Snap Ruler** switch. The closer vertex on the selected face is chosen instead of an interpolation of a position in the face (with the interpolation of the quantities as well).

8.6: Properties

8.6.1: Camera

The Camera Tab is shown in Figure 8-23. The Camera function controls the actual camera used to view the data.

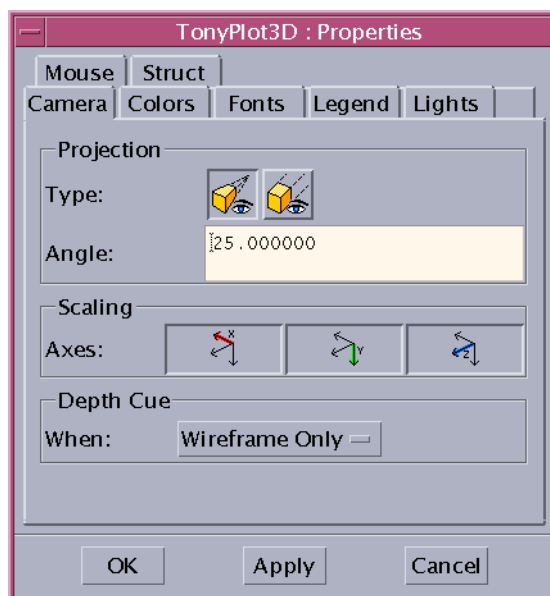


Figure 8-23: Camera Tab

Projection

This option controls whether or not the camera projects a perspective or parallel view. The angle of the projection can be changed when perspective is used

Scaling

This option effectively multiplies the coordinates of the structure to be viewed. You can set each axis that you wish to scale.

Depth Cue

Use this option to add more depth (realism) to the plot. Objects that are further away appear slightly dimmer than those that are closer. You can apply this option to the entire structure or just to the wireframe.

8.6.2: Color

This tab (Figure 8-24) can be used to adjust the colors of the various components of a structure.

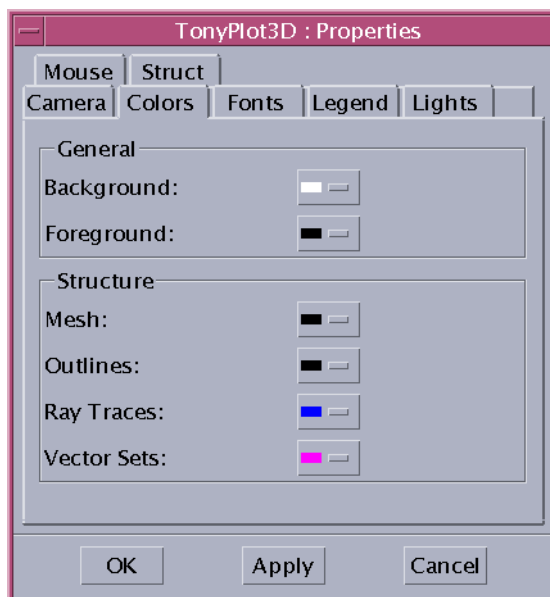


Figure 8-24: Colors Tab

8.6.3: Fonts

This tab (Figure 8-25) can be used to change the fonts used in plots.

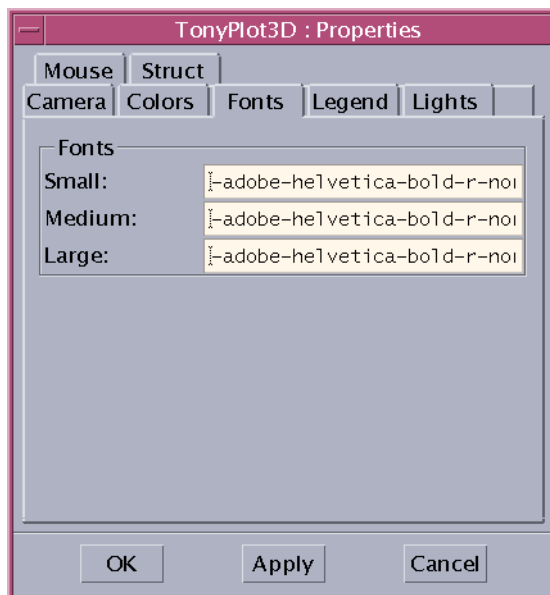


Figure 8-25: Fonts Tab

8.6.4: Legend

This tab (Figure 8-26) can be used to adjust the settings of the legends (Contours and Regions).

Note: The **Height** adjustment only pertains to the Contours Legend.

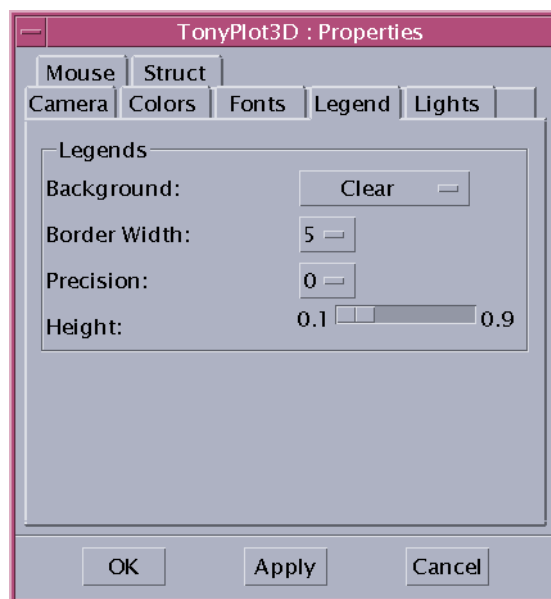


Figure 8-26: Legend Tab

8.6.5: Lights

The light sources in this tab (Figure 8-27) are used to illuminate the scene. In this tab, you can toggle the lights themselves (**Light 1** and **Light 2**), change the color of the lights, and modify their intensity. The directional light sources are described by latitude (vertical) and longitude (horizontal).

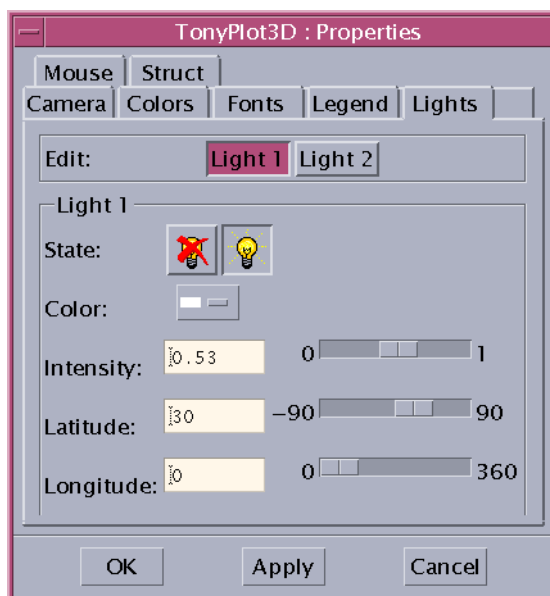


Figure 8-27: Lights Tab

8.6.6: Mouse

This tab (Figure 8-28) can be used to adjust the mouse buttons settings. See Section 8.3.4: “Plot Control Using the Mouse” for a description of these functions. **Automatic Movements** controls whether the structure will move automatically or not when you release one of the mouse buttons while still dragging the mouse.

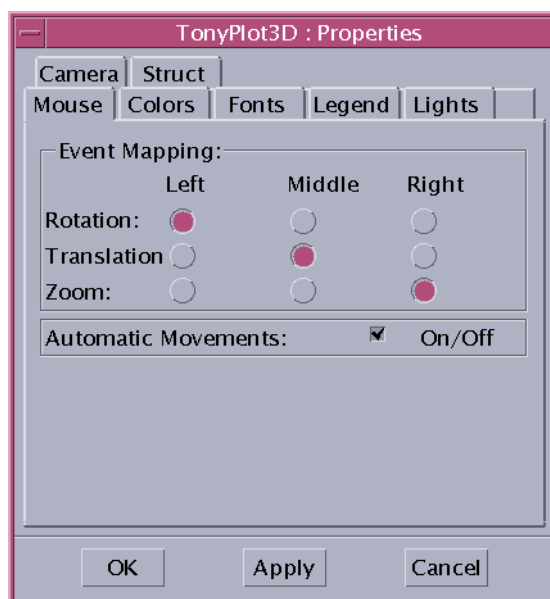


Figure 8-28: Mouse Tab

8.6.7: Structure

This tab (Figure 8-29) can be used to adjust the display properties of the structure. It contains the following:

- **Outline Controls:** To use this function, you have to plot your structure in **Solid & Meshed** Mode and use the Contours with a **Stepped** option in the Legend instead of a **Linear** one. When activated, it draws lines between steps on the elements.
- **Mesh Width:** Adjusts the width of the mesh.
- **Transparency:** Adjusts the transparency of the display.
- **Drag Type:** Defines whether or not a bounding box is drawn when the object is moved.

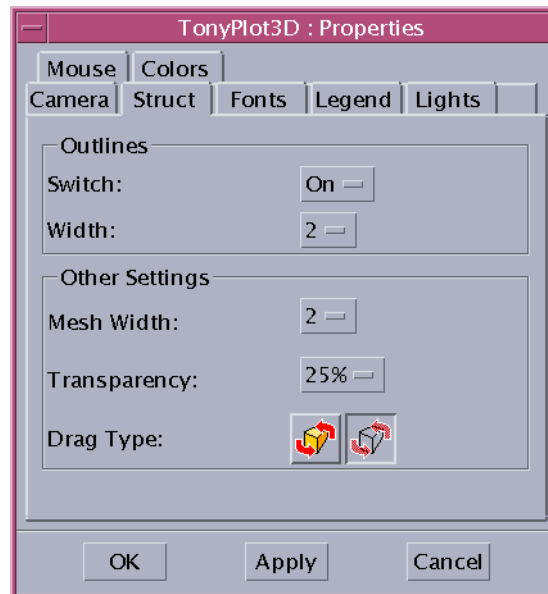


Figure 8-29: Struct Tab

8.7: Operating Platforms

8.7.1: SunOS 5.x UltraSPARC and SPARCstations

UltraSPARC and SPARC-based Systems

TONYPLOT3D executes on Sun UltraSPARC and SPARC-based systems supported by SunOS 5.7 or higher.

Operating System

This version of TONYPLOT3D requires SunOS 5.7 or higher. The command:

```
/usr/bin/showrev
```

shows which operating system version is running.

Graphics Hardware

There are two ways of producing screen images. One way is a software renderer, which uses graphics primitives implemented in software. Another way is a hardware renderer, which uses native graphics hardware implementing the OpenGL API version 1.1 or higher.

The software renderer works on UltraSPARC and SPARC-based models with True Color frame buffers and an X server. You can use `xdpyinfo` to obtain information about the X server on your machine.

To run TONYPLOT3D without graphics acceleration, use the `-nohw` command line option.

The hardware renderer requires a True Color graphics adaptor that supports OpenGL API version 1.1 (i.e., UltraSPARC and SPARC-based systems using the TCX, SX, GX, ZX, PGX/PGX24/PGX32, Creator/Creator3D, Elite3D, or Expert3D/Expert3D-Lite framebuffer).

Memory

TONYPLOT3D requires a minimum of 128Mbytes of real memory. To improve performance, however, more memory, such as 256Mbytes or even 512Mbytes, is strongly recommended. To see how much memory your system has, use the command:

```
/usr/sbin/prtconf
```

Memory size is printed within the first few lines of output.

OpenGL Library

TONYPLOT3D is designed to work with OpenGL API version 1.1 or higher. Try the following web site for information on obtaining OpenGL drivers for your graphics hardware:

<http://www.sun.com/software/graphics/opengl/>

or contact the Technical Support for your Operating System.

8.7.2: HP 9000/7xx Workstations

Workstation Models

TONYPLOT3D executes on any of the HP 9000 7xx series workstations that support HP-UX 11.0.

Operating System

TONYPLOT3D for HP platforms requires the HP-UX 11.0 operating system. The command:

```
uname -r
```

shows the current revision of the HP-UX operating system running on your workstation.

Graphics Hardware

There are two ways of producing screen images. One way is a software renderer, which uses graphics primitives implemented in software. Another way is a hardware renderer, which uses native graphics hardware implementing the OpenGL API version 1.1 or higher.

The software renderer works on workstations with True Color frame buffers and an X server. You can use `xdpyinfo` to obtain information about the X server on your machine. To run TONYPLOT3D without graphics acceleration, use the `-nohw` command line option.

The hardware renderer requires a True Color graphics adaptor that supports OpenGL API version 1.1 (i.e., the Visualize fx5 and fx10 graphics cards).

Memory

TONYPLOT3D requires a minimum of 128Mbytes of real memory. To improve performance, however, more memory, such as 256Mb or even 512Mbytes, is strongly recommended. To determine how much memory there is in your HP workstation, log in as "root" and enter this command:

```
/etc/dmesg
```

This generates a list of various system messages that were produced when your system was rebooted last. Look for a message like the one shown below.

```
Memory Information: Physical: 98304 KBytes
```

OpenGL Library

TONYPLOT3D is designed to work with OpenGL API version 1.1 or higher. Try the following web site for information on obtaining OpenGL drivers for your graphics hardware:

<http://www.software.hp.com/>

Then, choose **search**, **request OpenGL**, or contact the Technical Support for your Operating System.

8.7.3: Linux RedHat (PC Compatibles)

PC Models

TONYPLOT3D executes on any PC Compatibles that support Linux RedHat 7.2 or higher.

Operating System

This version of TONYPlot3D requires Linux RedHat 7.2 or higher.

Graphics Hardware

There are two ways of producing screen images. One way is a software renderer, which uses graphics primitives implemented in software. Another way is a hardware renderer, which uses native graphics hardware implementing the OpenGL API version 1.1 or higher.

The software renderer works on any PC compatibles with True Color frame buffers and an X server. You can use `xdpyinfo` to obtain information about the X server on your machine. To run TonyPlot3D, without graphics acceleration, use the `-nohw` command line option.

The hardware renderer requires a True Color graphics adaptor that supports OpenGL API version 1.1.

Memory

TONYPLOT3D requires a minimum of 128Mbytes of real memory. To improve performance, however, more memory, such as 256Mb or even 512Mbytes, is strongly recommended.

OpenGL Library

TONYPLOT3D is designed to work with OpenGL API version 1.1 or higher. Contact your graphics hardware vendor for information on obtaining OpenGL drivers for your graphics card.

8.7.4: Windows NT/2000/XP (PC Compatibles)

PC Models

Currently, TONYPLOT3D is not officially supported on Windows NT/2000/XP. But, TonyPlot3D will execute on any PC Compatibles that support Exceed or Exceed XDK version 7.1 (7.1.0.1 for Windows XP) or higher with the Exceed 3D add-on. Exceed 3D allows you to display OpenGL-based 3D applications using the Exceed X server by providing support for the GLX extension.

Operating System

This version of TONYPLOT3D requires Exceed or Exceed XDK 7.1 (7.1.0.1 for Windows XP) or higher with the Exceed 3D add-on.

Graphics Hardware

There are two ways to produce screen images. The first way is to use a software renderer, which uses graphics primitives implemented in software. The second way is to use a hardware renderer, which uses native graphics hardware implementing the OpenGL API version 1.1 or higher.

No software renderer has been provided by Silvaco for Exceed on Windows NT/2000/XP.

The hardware renderer requires a True Color graphics adaptor that supports OpenGL API version 1.1. You can use `xdpyinfo` to obtain information about the X server on your machine.

Memory

TONYPLOT3D requires a minimum of 128Mbytes of real memory. But for better performance and more memory, we strongly recommend 256Mb or 512Mbytes.

OpenGL Library

TONYPLOT3D is designed to work with OpenGL API version 1.1 or higher. Contact your graphics hardware vendor for information on obtaining OpenGL drivers for your graphics card.

Contact Hummingbird for information on getting Exceed 3D for Windows NT/2000/XP at www.hummingbird.com.

9.1: Overview

DEVEDIT is a device structure editor. It can be used to generate a new mesh on an existing structure, modify a device or create a device from scratch. These devices can then be used by Silvaco 2-D and 3-D simulators. DEVEDIT can be used through a Graphical User Interface (GUI) or as a simulator under DECKBUILD.

9.1.1: The Problem

A limitation of device simulators prior to DEVEDIT was inadequate or poor structure meshes. A mesh containing too many obtuse triangles or an insufficient number of triangles (too coarse a grid) may provide an inaccurate result or no result at all. A mesh containing too many triangles (too fine a grid) can result in excessive simulator processing time. Since the time most simulators use grows geometrically with the number of triangles (or grid points), it is critical to keep the number of triangles down to a reasonable number. Using simulators such as SSUPREM4 to create non-uniform meshes tend to be very time consuming and require a great deal of effort.

9.1.2: The Solution

DEVEDIT resolves these problems by allowing structures to be created or read into DEVEDIT in the form of Silvaco Standard Structure Files. The mesh contained in the file can then be replaced using the MESHBUILD algorithm. Refinement of the mesh is accomplished by setting parameters that describe critical areas or by simply pointing to the areas which require refinement.

In the process of creating a structure, definition of a device can be accomplished by simply drawing it on the screen. DEVEDIT can also perform analytic implants using built-in equations or cut lines from other simulators. Constraints are then placed on the mesh, to describe the critical areas of the device.

9.1.3: When to Use DevEdit

Use DEVEDIT when you want to perform the following operations:

Define a device interactively on the screen for subsequent device or process simulations.

- Remesh a device structure between process simulation and device test simulations, when the process simulator does not create a good grid for the device simulator.
- Remesh a device structure during a process or device simulation, when the mesh is no longer adequate for the next simulation step.

9.1.4: When Not to Use DevEdit

You should not use DEVEDIT to perform the following operations:

- Replacing numerical process simulations where accuracy is required.
- Meshing 1D device structures.

9.1.5: Getting Started

DEVEDIT can be run from the UNIX prompt or from DECKBUILD. There are two file types which DEVEDIT can read: Silvaco standard structure file format (common to all Silvaco simulators) and command format (a list of DEVEDIT commands which create a structure). The structure file format contains such information as triangles, impurity values, borders, etc. The command format is normally used when starting a device mesh from scratch. It contains the list of instructions that describe the current state of mesh development. Although the DEVEDIT command file is usually smaller than the structure file, it contains information, such as the state of DEVEDIT, impurity equations, and meshing modes.

Startup

You can start DEVEDIT one of two modes: 2D and 3D.

In 2-D mode, use:

```
devedit
```

In 3-D mode, use:

```
devedit3d
```

To start DEVEDIT in GUI mode, use one of the following commands for a UNIX prompt.

Command	Description
<code>devedit &</code>	This starts DEVEDIT with no active device.
<code>devedit fred.str &</code>	This starts DEVEDIT with the Silvaco Standard structure <code>fred.str</code> loaded.
<code>devedit fred.de &</code>	This starts DEVEDIT with the command file <code>fred.de</code> loaded.
<code>devedit3d &</code>	This starts DEVEDIT in 3-D mode with no active device.
<code>devedit3d fred3d.de &</code>	This starts DEVEDIT with the 3-D command file <code>fred.de</code> loaded. DEVEDIT does not currently support loading 3-D structure files.

Note: Make sure your DISPLAY environmental variable is set to an active X window screen.

9.2: Base Window

9.2.1: Layout and Functionality

The DEVEDIT base window display (Figure 9-1) is made up of several sections:

- **Control Buttons** - Displayed along the top of the screen, the series of menu buttons are used to control all DEVEDIT actions.
- **Main Canvas** - This area is used to show a graphical representation of the device.
- **Main Panel** - Displays a list of the regions in the current device and allows a region to be selected for region specific commands. The list contains all regions by name, number, legend, and (for color terminals) color.
- **User Added Impurities List** - Displays a list of the user-added impurities as they are selected from the Control panel Impurities menu, or read from a command file (impurities read from Silvaco standard structure files do not appear in this list). This list also allows impurities to be selected for impurity specific commands.

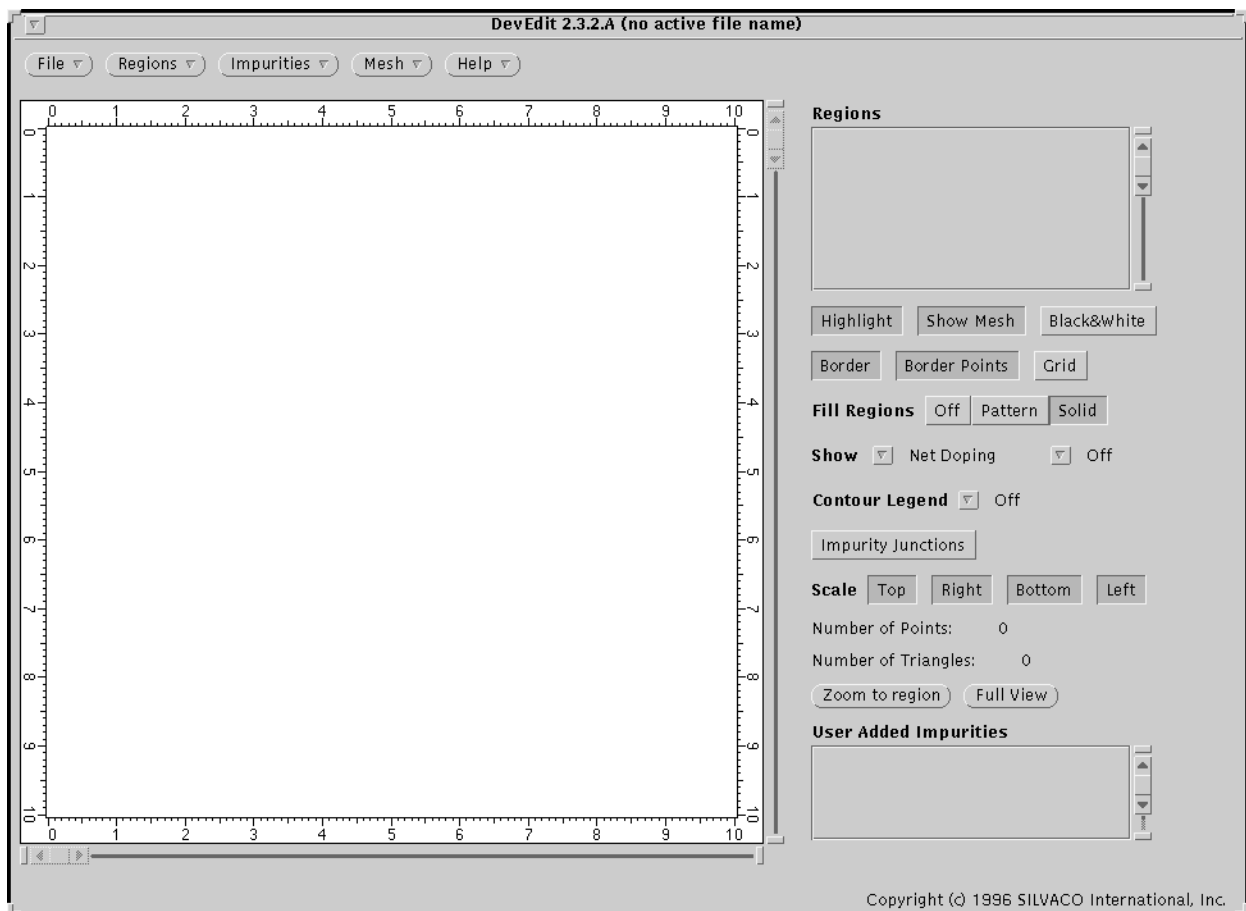


Figure 9-1:Base Window Display

9.2.2: Control Panel

Commands are performed by selecting the appropriate option from one of the menus or buttons displayed on the control panel as follows:

- **Files** - Contains a menu of load/save file operations.
- **Regions** - contains a menu of various material region control options.
- **Impurities** - Contains various options for control of impurities.
- **Mesh** - Contains a menu of various mesh control options.
- **Help** - Provides interactive help information.

9.2.3: Main Panel Controls

Controls located between the Regions panel and User Added Impurities panel control how regions and impurities are displayed. They perform the following functions:

- **Highlight** - Permits highlighting selected material region in red.
- **Show Mesh** - If a mesh exists, it can be shown or not shown to speed up display.
- **Black & White** - Allows making a black and white picture on a color screen.
- **Border** - Shows a black border around regions. It can be turned off so regions are not separated by a border.
- **Border Points** - Shows key points used to create the mesh.
- **Grid** - Shows intersect points between scales on main panel.
- **Fill Regions** - How regions should be filled in.
- **Off** - Regions are not filled.
- **Pattern** - A fill pattern is used to fill region. Ordinarily used only in Black & White mode.
- **Solid** - The region's color is used to fill region. This option is not shown on Black & White terminals.
- **Show Net Doping** - Permits selection of how doping is displayed (normally, coarse or off is selected).
- **Net Doping Legend** - Permits positioning of a legend in one of eight positions on the main panel.
- **Impurity Junctions** - Shows where the junctions are. (Only shown when a mesh exists.)
- **Scale** - Permits display (or not) of scale on each border of the main panel.
- **Number of Points** - Displays number of points in current structure.
- **Number of Triangles** - Displays number of triangles in current structure.
- **Zoom to Region** - Permits selection of a region on the Regions list as full view on main panel.
- **Full View** - Selects full view of work area (entire device).

9.2.4: Control Windows

During the process of DEVEDIT operation, a number of auxiliary panes can stack up obscuring the Main panel. To regain access to the Main panel, sequentially remove the auxiliary panes by choosing the **Cancel**, **Done**, or **Apply** button (as appropriate) for each panel.

9.3: FILE CONTROL

9.3.1: Using Devedit

Using DEVEDIT is largely intuitive. As a starting point, a screen appears displaying a space where you can edit a device. Menus and options appear on the right hand side of the screen. They are selected from the **Main Menu** button options along the top of the main screen. Each action may change the control panel on the right side of the main screen. Select the required menu options from the series of buttons and menus, and press **Apply** or **Done** on the control panel when complete.

9.3.2: Loading a Silvaco Standard Structure File

A Silvaco Standard Structure file can be loaded at any time into DEVEDIT. A Silvaco Standard Structure File may have been created by any 2-D simulator. A Silvaco standard structure can be loaded from the time DEVEDIT was invoked, (see Invoking DEVEDIT from the UNIX prompt) or by using the **File menu**. Under the **File menu** button, on the left side of the DEVEDIT screen, a number of menu options allow the basic Silvaco standard structure file I/O control of DEVEDIT. To load a file, select the **Load menu** option. A popup window appears indicating a **Directory name**, a **filename** and a **Filter type**. Two types of file can be loaded into DEVEDIT. First, a **Silvaco Standard Structure File** that contains a description of the entire structure to be loaded. Second, a **Command file** which contains the a complete list of instructions developed as operations performed to create a structure. If you are starting from a 2-D simulator only, you will not have a **Command file** to load. If starting from a point previously run in DEVEDIT, a **Command file** may exist. **Command files** can be distinguished by the filename extension .de.

To load a **Silvaco Standard Structure File**, pull down the **File** menu and choose **Load....** In the new window, set the filter to *.str unless you do not end all your **Silvaco Standard Structure Files** with .str. Set the current directory and file name if need. Currently, the mesh in the **Silvaco Standard Structure File** is ignored. Therefore, you must put on a new mesh before saving the structure file.

9.3.3: Saving a Silvaco Standard Structure File

DEVEDIT can save both structures and **Command** files. Saving a structure means the object displayed on the screen is saved to a **Silvaco Standard Structure File** (*.str). This is done only after a mesh has been generated. Saving a command file means that the complete sequence of events required to arrive at the current structure are saved while in DEVEDIT. The command file is a complete object oriented history of events required to create the current structure. To save a structure file, click on the **Silvaco Standard Structure File** option before clicking on the **Save file** button. To save a Command File, click on the **Commands** option button before clicking on the **Save file** button. The files are saved to the directory from where DEVEDIT was invoked. A structure can be incrementally saved to a given filename by clicking left on the **File** button. This is equivalent to selecting the **Save** option under the **File** menu.

9.3.4: Difference - Silvaco Standard vs. Devedit

A **Silvaco Standard Structure File** and a DEVEDIT file (Command) differ. A **Silvaco Standard Structure File** constrains some information about each region, a mesh describing the device, and a list of impurities at each point. This file type must be used to pass the structure to a simulator. A DEVEDIT file contains the DEVEDIT cards (instructions) needed to create a device, the analytic impurities functions, mesh creation cards, and DEVEDIT mode settings. It does not currently support impurities (including doping and results) read in from a structure file. Therefore, if a structure file is read in, a structure file must be written out, or all the impurities not added during the current editing session are lost. Subsequent releases will rectify this problem.

When starting a device mesh from scratch, it should be saved in both card deck and structure file formats. When saved in the command format, all original information is retained such as formulas, region colors, mesh constraints, etc. Structure files are necessary for use by ATLAS. However, when a file is saved in the structure file format, only the resulting values of calculations are retained, pre-empting the possibility of future modification of the original DEVEDIT commands.

9.3.5: Loading a Command File

Pull down the File menu and select **Load...** In the new window, set the filter to **LOADING A COMMAND FILE - *.de]**.de** unless you do not end all your DEVEDIT Command files with .de. Set the current directory and file name if needed.

9.3.6: Default Files

In addition to SILVACO Standard Structure Files and DEVEDIT Command files, there are also DEVEDIT Default Option files. These files are used by DEVEDIT to reset DEVEDIT options when loading a SILVACO Standard Structure File or when starting DEVEDIT with no file. If loading a DEVEDIT command, most options return to the active settings when that command was saved.

9.4: Tutorial

This tutorial discusses aspects of mesh creation in DEVEDIT. It begins by discussing the goal of an efficient mesh for device simulation. There are two examples. The first example demonstrates how to create a structure. The second example illustrates how to create a mesh in an existing structure. These two examples illustrate basic usage of DEVEDIT, but not all features are discussed in this tutorial. We recommend that you read both examples. Finally, in this tutorial, some advanced features are mentioned. The remainder of the DEVEDIT chapter should be used as a reference.

9.4.1: Goal And Purpose Of Creating A New Mesh

Specifying a good mesh is a crucial issue in device simulation. There is a trade-off between the requirements of accuracy and numerical efficiency. Accuracy requires a fine mesh that resolves the structure in solutions. Numerical efficiency is greater when fewer points are used. The critical areas to resolve are difficult to generalize, since they depend on the technology, transport phenomena, and bias conditions. A generalization is that critical areas tend to coincide with reverse-biased metallurgical junctions. Typical areas that require fine mesh includes:

- high electric fields at the drain/channel junction in MOSFETs.
- the transverse electric field beneath the MOSFET gate.
- recombination effects around the emitter/base junction in BJTs.
- areas of high impact ionization.
- around heterojunctions.

The cpu time required to obtain a solution is typically in proportion to

N^a

where N is the number of nodes and a varies from 2 to 3, depending on the complexity of the problem.

Thus it is most efficient to allocate a fine grid only in critical areas, and a coarser grid elsewhere. Additional factors to consider in a mesh are:

- avoid obtuse triangles in semiconductor regions, particularly in current path and high field areas.
- avoid abrupt discontinuities in mesh density.
- avoid thin triangles, the ratio of longest to shortest edge in a mesh triangle should be on the order of 10, but not 100.
- use several mesh layers in a material layer, particularly for very thin material or doping layers.
- for most simulations, 2000 - 3000 mesh points are adequate.

These principles can generally ensure accurate solutions with quick convergence times. Poor meshes can lead to inaccurate answers, poor convergence times or even lack of convergence, leaving you without a solution, and causing frustration. A significant number of device simulation problems are caused by not adhering to the above principles.

9.4.2: EXAMPLE 1 - CREATE A NEW STRUCTURE

To start DEVEDIT, enter:

```
devedit &
```

in a terminal window. This starts DEVEDIT in the GUI (opposed to batch) mode. Note the work panel on the right side of DEVEDIT, this home panel returns after other functions have been applied. In this first example, a recessed AlGaAs/InGaAs HEMT is created.

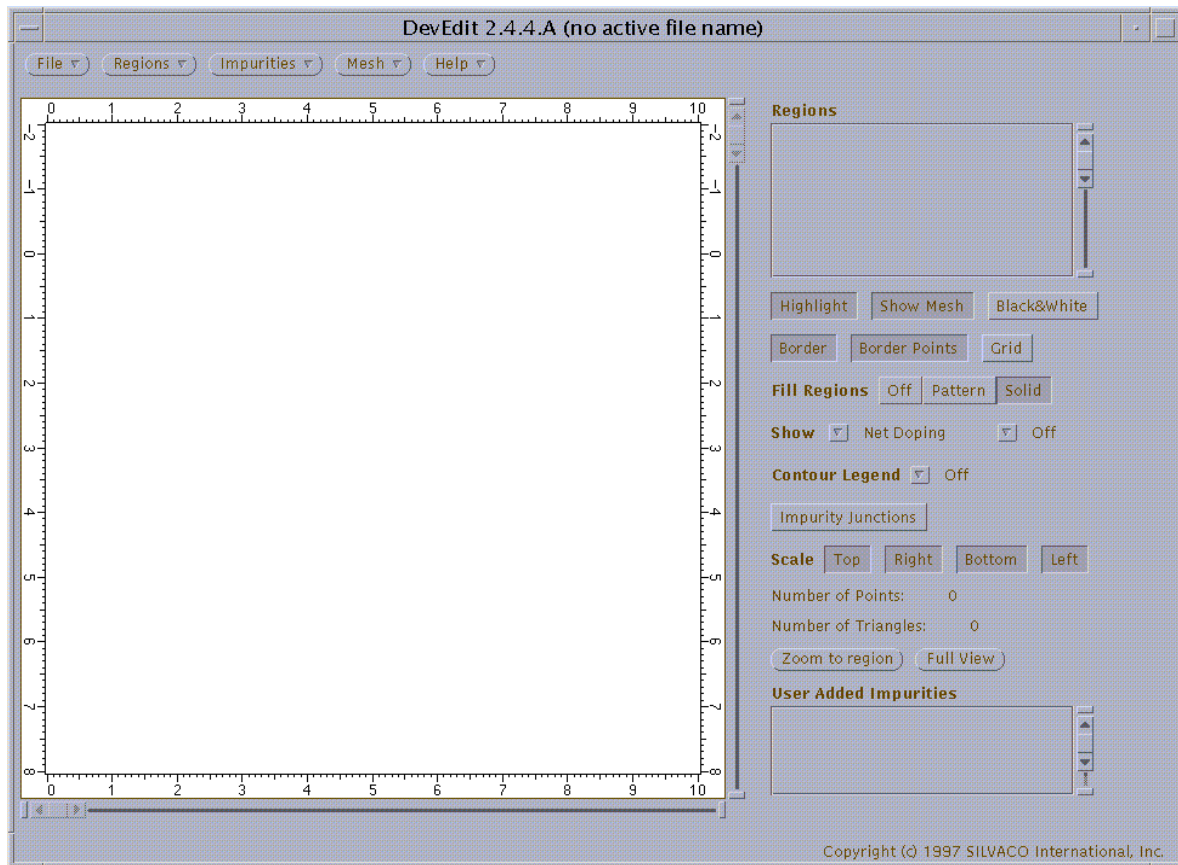


Figure 9-2: DevEdit Graphics User Interface

Work Area

DEVEDIT starts with a work area that uses a default setting. To change the size of this work area, use the Menu (right) mouse button over **Regions** and select **Resize Work Area...**, and a new panel is then displayed on the right. Change the Depth (y) and Length (x) minimum and maximum can be set to desired values. For this tutorial, set y_{\min} to -0.05, y_{\max} to 0.5, x_{\min} to zero, and x_{\max} to 4.5; press Return after typing each value. Click on **Apply**. Careful choice of the device in the coordinate system can prevent confusion at later stages.

Note: After entering values into the text boxes in DEVEDIT, press Return for the value to be accepted.

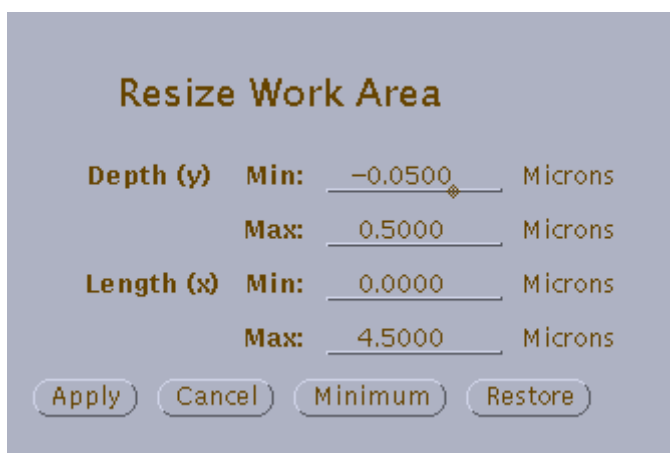


Figure 9-3: Resize Work Area Menu

Defining Regions

Adding a Region

Creating a device begins with adding regions. Each region consists of one material, although an area of a single material can consist of more than one region. Under the **Regions** pull-down menu, choose **Add region....** The menu on the right of DEVEDIT changes, with the title **Add Region**. DEVEDIT allows for geometrical shapes to be created, using the mouse or by typing the coordinates of the region.

To begin, create the entire AlGaAs substrate region:

Using the Mouse

Use the left mouse button and click on the location (0, 0.05), noting that the location of the mouse is displayed in the x and y locators. Continue to choose the points (4.5, 0.05), (4.5, 0.5), (0, 0.5). If a mistake was made in the locations of the points, press the middle mouse button to remove the point.

Using the Keyboard

Alternatively, type the x and y locations of each point in the bottom of the right panel, and press return after each number, and then click on **Add**. The point is then be displayed in the work area. It is also possible to change the location of a point by selecting the point in the Polygon box (using the left mouse button, so the point is highlighted), change the x or y locations by entering the correct location, pressing return, then clicking on **Replace**.

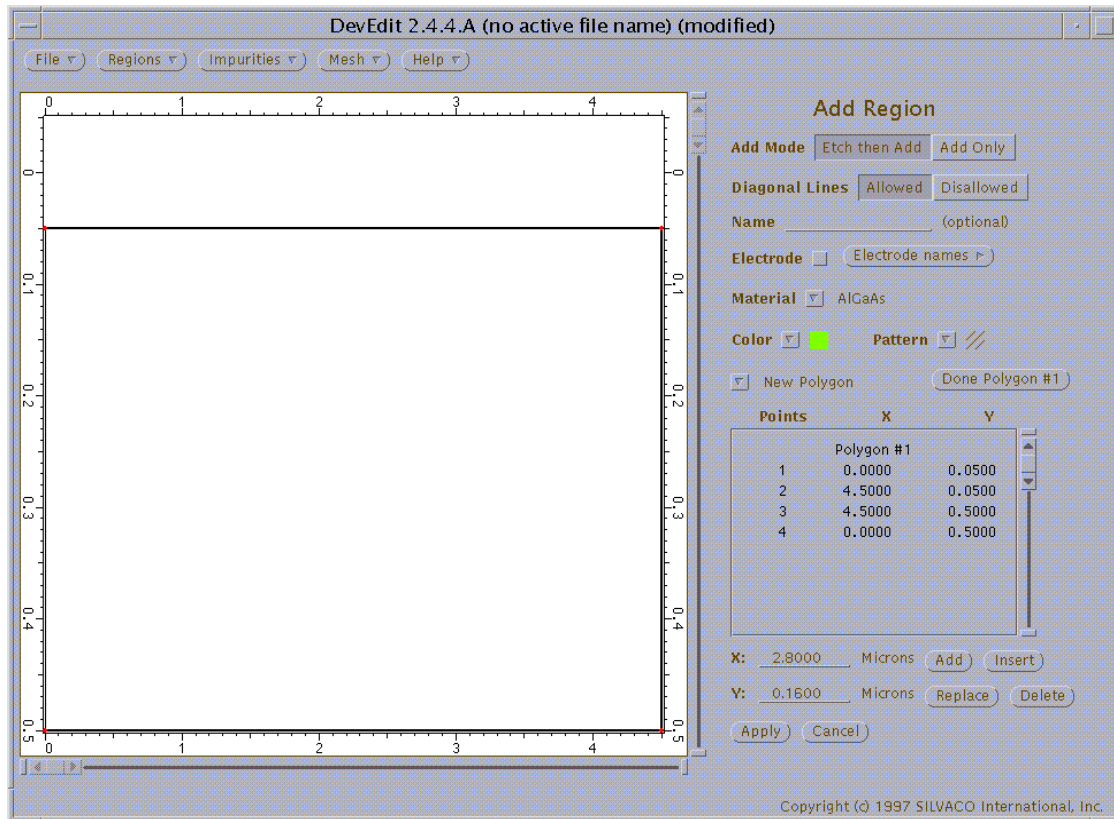


Figure 9-4: Add Region and Constructing a Device

Material Selection and Uniform Doping

After entering the corners of the region, the material and doping need to be entered.

Use the right mouse button at the **Material** menu button, choose AlGaAs. Finally, using the right mouse button at the **New Polygon**, select **Set Base Impurities**, then at **Doping Type**, select **Generic Donors/Acceptors**, and enter $1e14$ by **Acceptors**: then press Return.

Add Region

Add Mode

Diagonal Lines

Name _____ (optional)

Electrode ☐

Material ▾ AlGaAs

Color ▾ **Pattern** ▾

▾

Doping Type ▾ Generic Donors/Acceptors

Acceptors:

Donors:

Net Doping:

Figure 9-5: Adding Uniform Doping Throughout a Region

Setting Mole (Composition) Fraction

Composition fractions for ternary and quaternary materials are defined in the ATLAS USER'S MANUAL, Appendix B.

In order to enter the composition fraction for a material in DEVEDIT, again go to **Set Base Impurities**, and under **Doping Type**, select **Composition Fractions**. Enter 0.3 for the **Comp. Fraction X:** and press Return. The AlGaAs substrate region is now completely defined, with geometry, doping and composition fraction defined. Once done, click on **Apply**.

▾

Doping Type ▾ Composition Fractions

Comp. Fraction X:

Comp. Fraction Y:

Figure 9-6: Setting the Mole Fraction

Modifying Regions

If you want to change a region, correct a mistake, or have clicked on **Apply** before all the settings (including doping and composition fraction) have been entered, the easiest method to correct the problem is to modify the existing region. When DEVEDIT is displaying the home menu on the right, simply left click on the region name in the upper right box, then right click the **Region** pull-down menu, and select **Modify Region**. The right display lists the existing information in a similar format to **Add Region**. When modifications are complete, click on **Apply**.

Next, add the GaAs cap regions. Again, under **Regions**, select **Add Region**. Enter a polygon beginning at (0,0). Continue to choose the points (1.1, 0.0), (1.5, 0.05) and (0.0, 0.05). To add doping, select **Generic Donors/Acceptors**, and enter 5×10^{21} by **Donors**.

Enter another polygon in the same manner to create the right cap region, at locations (2.9, 0), (4.5,0), (4.5, 0.05), (2.5, 0.05), GaAs, with the same doping.

Etch then Adding a Region

Next, an InGaAs layer is added. This step is performed in a similar way to adding a region, except the location is being etched from existing material (the AlGaAs substrate). The default setting for **Add Mode** is **Etch then Add**, meaning the new region displaces, or overwrites, any existing material region. The other mode, **Add Only**, adds a new region only where no other region has been defined. Add an InGaAs region from (0, 0.084), (4.5, 0.084), (4.5, 0.098), and (0, 0.098). The precision of these coordinates exceeds the default of the mouse setting. You have two options, either enter the coordinates of each point and click **Add** after each point, or use the mouse first to select approximate locations, then select the points in the **Polygon** box, and edit the points (click **Replace** after each point). Set the doping to 5×10^{15} donor concentration.

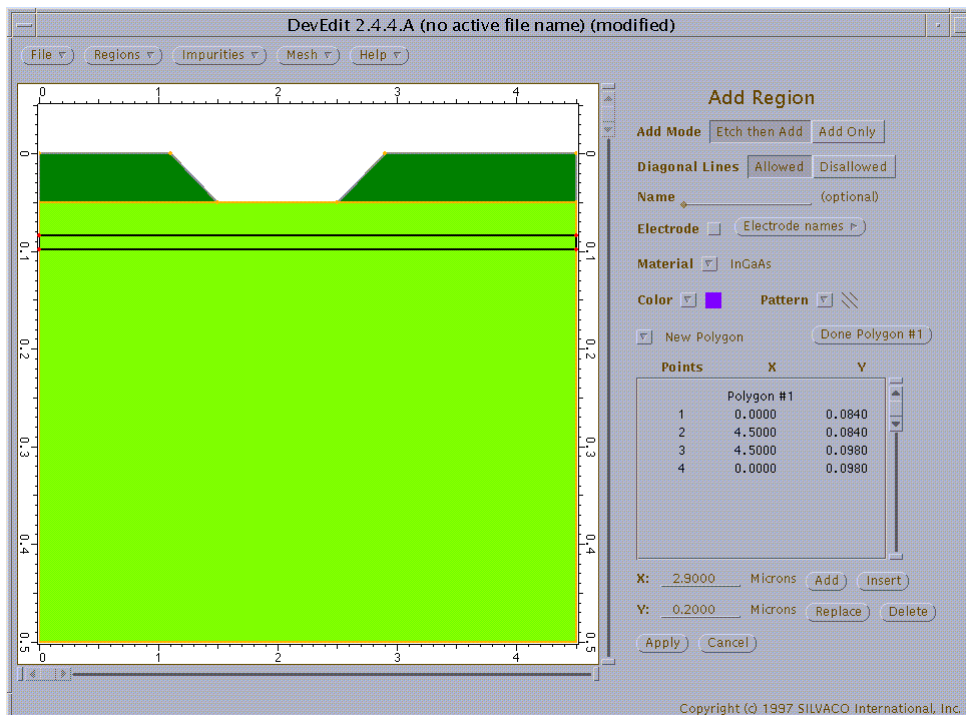


Figure 9-7: Etching, then Adding a Region - Overwriting a Material in an Existing Area

Add Electrodes

The last regions to add are the electrodes. In ATLAS, only the boundary of an electrode contacting the semiconductor is considered, hence the height of the electrode and the mesh inside of the electrode are of little consequence.

Add a gold region at (1.7, 0), (2.3, 0), (2.3, 0.05), (1.7, 0.05). In order to identify this region as an electrode, click on the button next to **Electrode**, then use the right mouse button to click **Electrode names**, and choose **gate**.

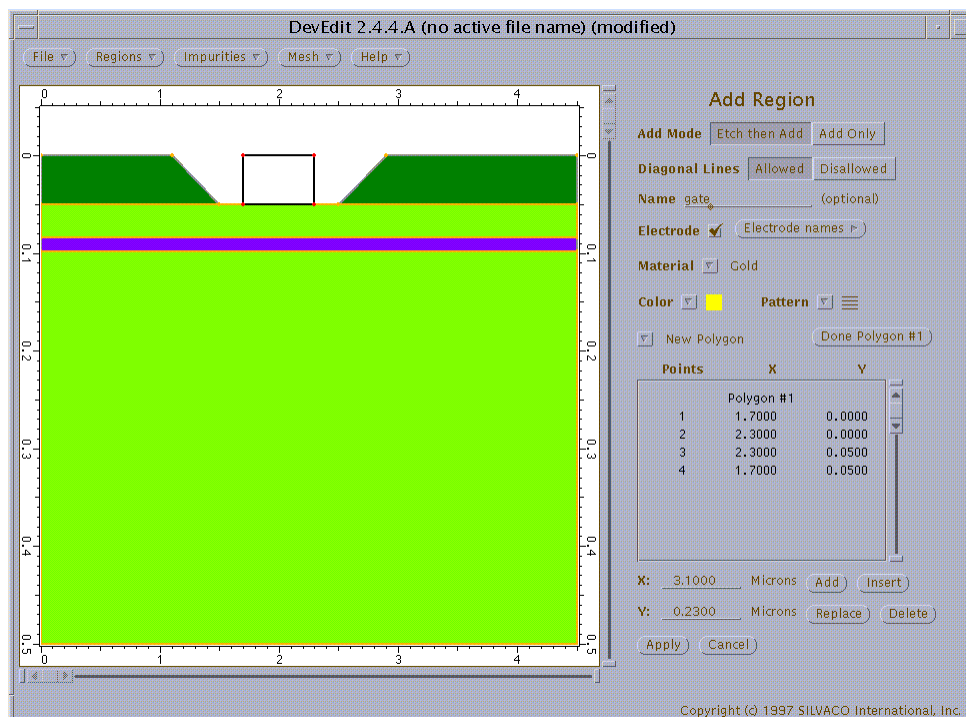


Figure 9-8: Adding a Gate Electrode

In a similar manner, add a source contact at (0, 0), (0, -0.01), (1, -0.01), and (1, 0). Add a drain contact at (3.5, 0), (3.5, -0.01), (4.5, -0.01), and (4.5, 0). Much of the geometry of the device is now complete. The last step is to add impurities.

Defining Impurities

Adding Impurities

If an impurity concentration is constant through a region, the preferred manner to define the dopant is in the **Region** definition, as discussed above. For a non-uniform impurity, including impurities that are in more than one region, then the impurity should be separately defined, as described below.

Under **Impurities**, select **Add impurity...**; the menu for impurities becomes available. Next to **Impurity**, select **Donors**. For Start x and Y values, use (0,0) and (1, 0.08). Note that only two points are selected, not four, and the opposite corners of a rectangle should be selected. (A one-dimensional line is acceptable; it is considered a flat rectangle). This region is the impurity source region, and the impurity concentration is the peak concentration, uniformly throughout the region. Set the **Peak Concentration** to $5e+21$ and the **Reference Value** to $1e+20$. The **Reference Value** is used as a scaling factor in the Roll-Off Functions (see Section 9.12: "ROLL-OFF FUNCTION").

The **Roll-Off** functions calculate the vertical and horizontal roll-off from the source impurity region (the peak concentration), as a function of x , y , and in three dimensional structures, z . The **Join Function** calculates the dopant concentration when both an x and y distance exists. Set the **Join Function** to **Multiply**. Select the **Y Roll-off** as **Gaussian (Dist)** and **Distance: 0.048**. Select the **X Rolloff** as **Error Function and Constant: 0.02**. Click on **Apply**. Remember to press Return after entering the numbers.

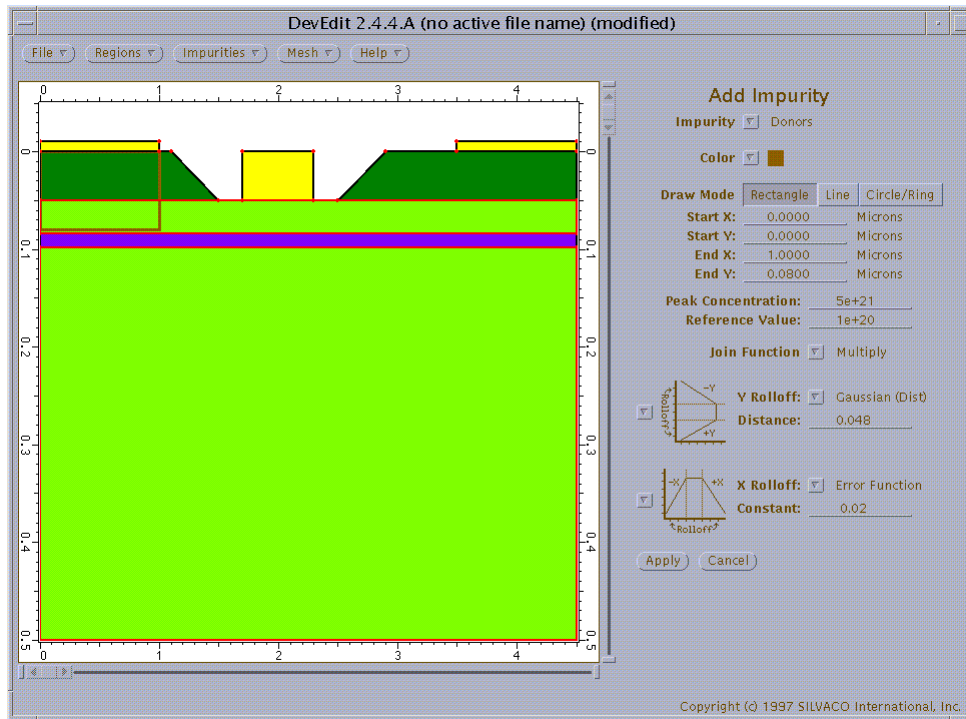


Figure 9-9: Add Impurity

Add another impurity region from (3.5, 0) to (4.5, 0.08) with similar roll-off properties.

Displaying the Doping

Back on the main menu, there are options to display, or **Show**, attributes of the structure. The first button defaults to **Net Doping**, and the second button defaults to **Off**. Select the second button to **Fine**. This option shows a relatively fine gradation of doping throughout the structure. **Coarse**, **Medium**, and **Very Fine** display options exist also, but be alert that the finer the display, the longer the refresh time takes. **Options** under **Net Doping** includes separate options for donors, acceptors, and specific impurities. One can also select the location of the **Contour Legend**.

Modify Impurities

The impurity regions added are listed in the main menu, at the bottom under **User Added Impurities**. Select the impurity of interest (left mouse button), then under the **Impurities** button, select **Modify impurity...** A new menu, similar to the **Add Impurity** menu, becomes available. Using these principles and techniques, you can design any device of interest. The next concern is creating a mesh for the device.

Mesh Creation

MeshBuild

The **Mesh** pull-down menu has a **MeshBuild** command. This command reads the current boundary conditioning conditions, the base mesh, the geometry and impurities, and mesh constraints, and creates a mesh, using the mesh parameters available. The default parameters create a mesh adequate to define the geometry of the device, but little else. It is unsatisfactory to describe the details of the impurity distribution, for instance, or the material layers for device simulation in ATLAS.

Mesh Parameters

Under the **Mesh** pull-down menu, there is a command of **Mesh Parameters**. Base Mesh Height and Width may be used to create more symmetric meshes for some devices. For most devices, setting Mesh Constraints provide the same results, and also allow more fine tuning. Therefore, these parameters were not used in this example. In devices with extremely detailed region borders, it may be turned **Off**. Click on **Cancel**. (See Section 9.9.2: “Boundary Conditioning” is discussed in the next example).

Refine on Quantities

This command allows you to refine the mesh on gradients of various quantities, including donors, acceptors, total and net doping, molar composition. (If the structure was imported from ATLAS, other quantities including electrical field, and potential, are available). This is a useful tool for refining the grid on areas that require a finer grid, namely, where gradients exist.

Right click on the **Add** button, and select **Donors**. Then, right click on the **Mesh** pull-down menu, and select **MeshBuild** again. **MeshBuild** operates by building a mesh with instructions of mesh generation, which have been modified when the **Refine on Quantities** was changed to include donor gradients.

Note: Note that the mesh has changed to include a finer mesh where donor concentration gradients exist.

Click on **Done** at the bottom of the panel. This action returns you to the main DEVEDIT panel. (If the existing panel is not the main panel, click on **Cancel** or **Done** on the existing panel. DEVEDIT returns you to the previous panel, and ultimately to the main panel). Near the bottom of the main panel the **Number of Points** and **Number of Triangles** are listed. These numbers are useful, in order to gauge the total number of points with the more subjective interpretation of the quality of the grid.

You can go back to **Refine on Quantities**. The **Scale** defaults to Logarithmic, which is appropriate for dopant concentrations. Change the **Sensitivity**, then rebuild the mesh (**MeshBuild**), and observe the effect on the total number of points. The **Sensitivity** setting controls the extent of the gradient in which mesh points are added. A higher setting reduces the density of the mesh. A lower setting increases the mesh density. The **Transition** value is the minimum value that is considered for Meshbuilding purposes (i.e., example, a gradient donor concentration from 10e9 to 10e8 is not considered when the **Transition** value is set to 10e10 (the default)).

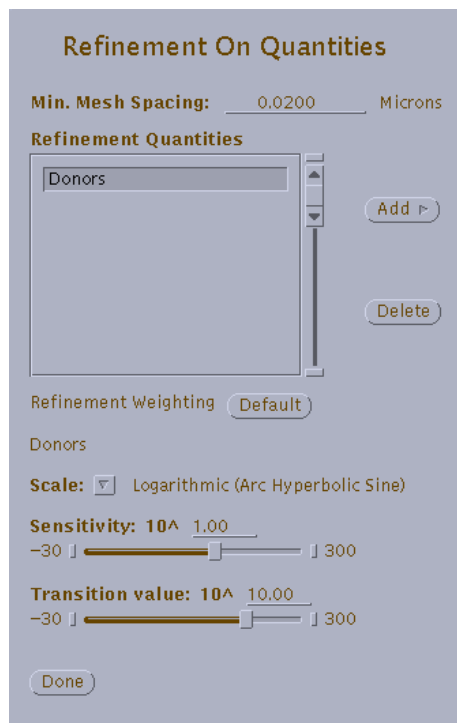


Figure 9-10: Mesh Refinement on the Donor Gradient

Mesh Constraints

The **Mesh Constraints** section is the principal area for controlling mesh construction. In this section, you can control the maximum triangle ratio, the maximum and minimum height and width, either throughout the device, or selectively in given regions, in material types, or underneath regions or materials.

Begin with selecting **Semiconductor Regions** in the **Material Types and Regions** box. Click on the box next to **Max. Height**, and set the value (either by typing or moving the slide bar) to 0.05 (all units of distance are microns). Similarly, set the **Max. Width** to 0.25. These values are valid for the mesh creation in all semiconductor regions. As mentioned previously, the mesh in insulators and electrodes are unimportant, so it is acceptable to leave the mesh arbitrarily large in these regions. By clicking on **MeshBuild** again, you will notice the mesh in the semiconductor regions now adheres to these criteria.

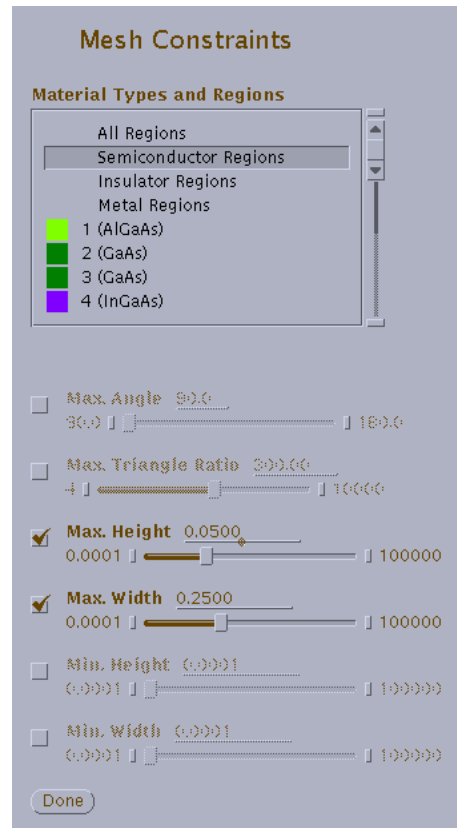


Figure 9-11: Mesh Constraints Menu

Select the upper AlGaAs region. Again, the same mesh criteria are available for control for this single layer. Change the **Max. Height** to 0.01. It is strongly recommended that each material layer, or region, have several (such as four) mesh layers. You can also adjust parameters for other layers. It is recommended in this example to increase the mesh density in the semiconductor layers such that several mesh layers exist in each region.

Manual Refine Box

Also under the **Mesh** pull-down menu is the **Refine Box**, in which the **Refine X**, **Refine Y**, **Refine Both**, and **Unrefine** options exist. This meshing does not work within **MeshBuild**, but rather doubles the density of the mesh, in the area specified, in the X, Y, or both directions, or reduces the mesh density by half. However, because these mesh changes are not easily reproduced, this action is not recommended.

Saving The File

The file can be, and should be, saved in two formats; the structure file, and the command file. The structure file is a format used by other Silvaco programs, including TONYPLOT and ATLAS, so it is necessary to save the structure file for continued device simulation. The command file is a list of the instructions used by DEVEDIT to create the structure and the mesh. To make additional changes at a later time, save a command file so that the original set of **DevEdit** instructions can be read. Additionally, running **DevEdit** in batch mode requires the command file.

Structure File

By right clicking on **File**, select **Save as...** and a pop-up menu is displayed. By convention, Silvaco structure files end with the extension **.str**. Choose a name and use this extension (for instance, **example1.str**), then press the **Save Structure** button. The file can now be used in ATLAS and TONYPLOT.

Command File

In a similar manner, select **Save as...**, and enter a name ending with the extension `.de` (for DEVEDIT). It is good practice to use the same filename as the structure file, but ending with the conventional `.de` extension, hence `example1.de` for this case.

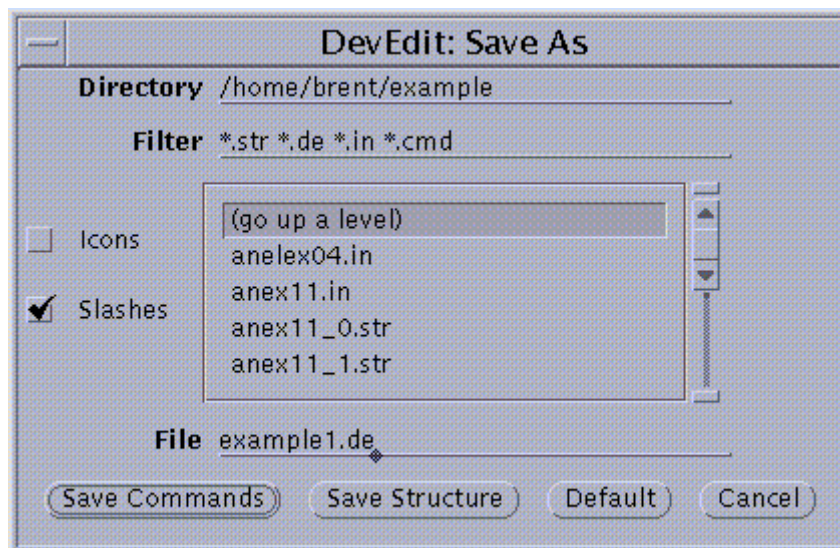


Figure 9-12:File Save Menu

Batch Mode

It is also possible to run DEVEDIT in batch mode, within DECKBUILD, without using the graphics user interface (GUI) mode used above. Although a user could directly input the command statements to create a structure and mesh, this method is more difficult than using DEVEDIT in GUI mode. However, once the command file has been saved, it can be easily loaded, edited, and run in batch mode within DECKBUILD.

Start DECKBUILD with `deckbuild &` then left click on **File (Open)**, and select a DEVEDIT command file (`example1.de`). This command file can be run inside DECKBUILD, with minor modifications.

1. The first line of the DEVEDIT is similar to `DevEdit version=2.4.0.R` in which it is recommended that this line become a comment, beginning with `#`. Then, a new first line should be added that reads: `go devedit` or if you want to retain the version number `"go devedit simflags="-V 2.4.0.R"` (or whatever the version number of DEVEDIT in use).

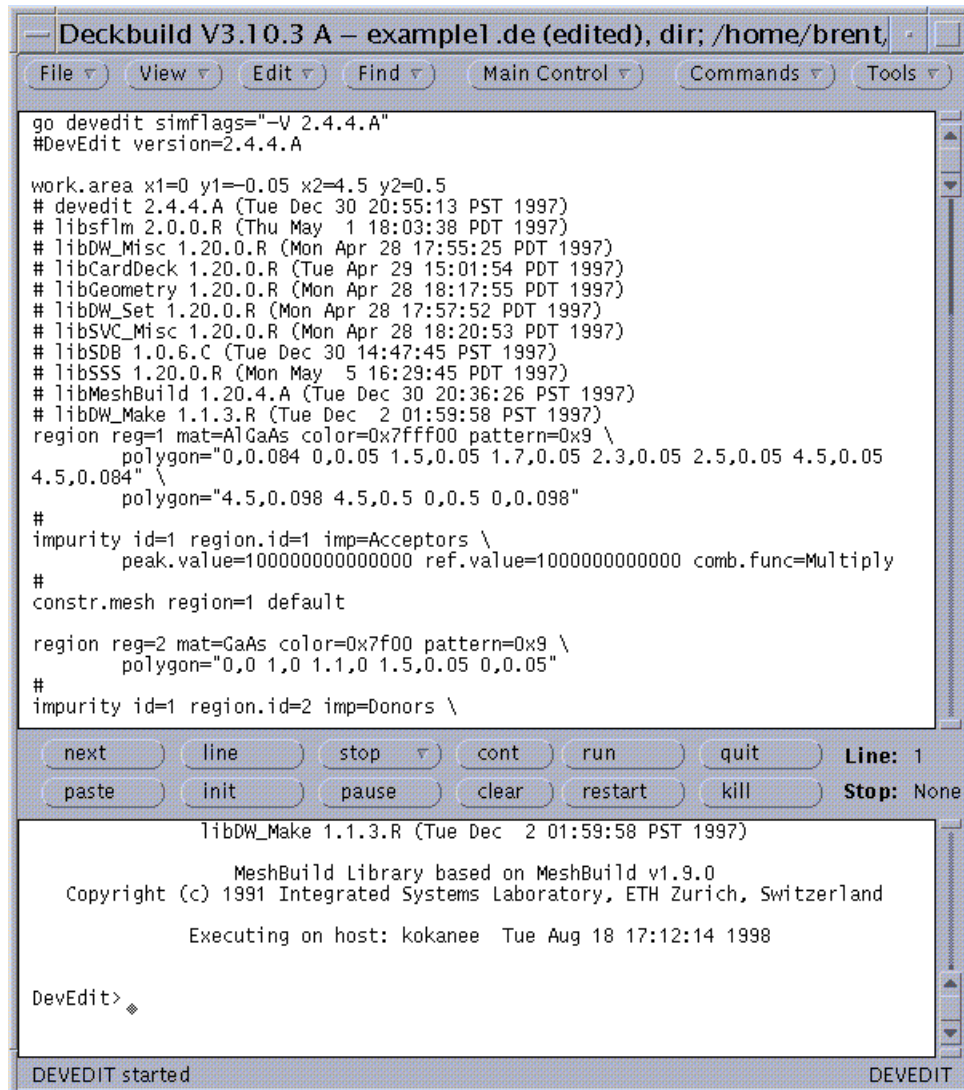


Figure 9-13:Running DevEdit in Batch Mode within DeckBuild

2. We also recommend that you add the line structure outfile=test.str at the end of the file to explicitly save the structure in a structure file.

9.4.3: EXAMPLE 2 - REMESHING AN EXISTING STRUCTURE

Loading The Structure

Typically, a structure is created in ATHENA, and saved in a structure file. For this tutorial, an existing Silvaco example will be used.

Obtain Existing Structure

Start DECKBUILD, by entering:

```
deckbuild &
```

in a UNIX terminal window. Right mouse click on **Main Control**, then select **Examples....** The DECKBUILD: EXAMPLES library of examples is then displayed in a separate pop-up menu. Double-click on MOS1, then double-click again on mos1ex01.in. Then click on **Load example**, which loads the ATHENA/ATLAS input deck into DECKBUILD, which copies the structure and log files into the directory where you launched DECKBUILD.

In DEVEDIT, left click on **File**, then select **Load**. Then highlight (again with the mouse) mos1ex01_0.str, then click on **Load File**. The ATHENA created structure is now in DEVEDIT. DEVEDIT command files (*.de) are loaded in the same manner.

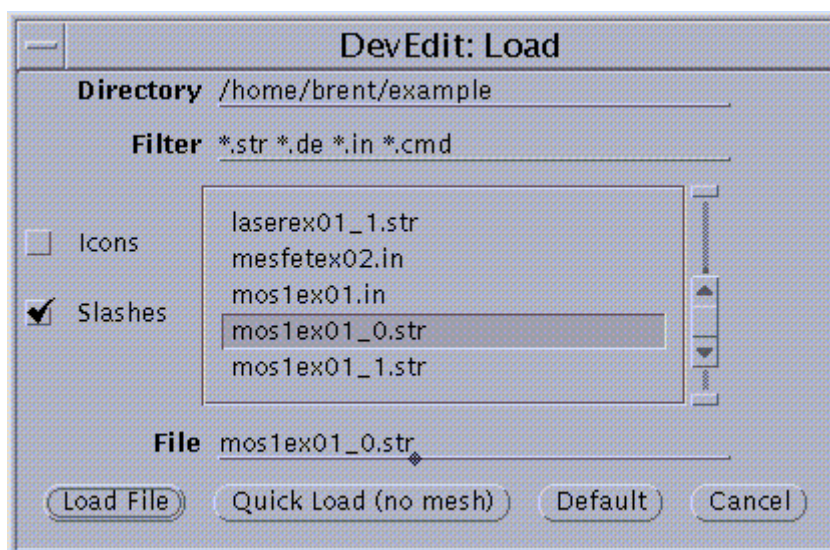


Figure 9-14: DevEdit Load File Menu

Structure Editing

It is recommended that if the structure was made in ATHENA, that no changes to the structure be made (i.e., material boundaries, doping distribution). If the structure was created in DEVEDIT, then the command file (*.de) should be edited, but not the structure file. Upon completion of edits to a DEVEDIT command file, both a command file and a structure file should be saved.

For this example and this purpose, only the mesh should be changed. Minor changes to the material boundaries will be done in boundary conditioning, described below.

Display

Zoom

You can depress the left mouse button anywhere on the displayed structure, and drag it to another location, defining a rectangle between the two points. Upon releasing the mouse button, only the selected area is displayed (zoom). The bottom and right axes will have sliders for panning. The **Full View** button on the right menu return the display to the entire work area.

Displaying the Doping

Also on the main menu, there are options to display, or show, attributes of the structure. The first button defaults to **Net Doping**, and the second button defaults to **Off**. Select the second button to **Fine**. This option shows a relatively fine gradation of doping throughout the structure. **Coarse**, **Medium**, and **Very Fine** display options exist also, but be aware that the finer the display, the longer the refresh time takes. **Options** under **Net Doping** include separate options for donors, acceptors, and specific impurities. One can also select the location of the **Contour Legend**, by using the right mouse button adjacent to **Contour Legend**.

Impurity Junctions

This toggle switch will display the p-n junctions in a device.

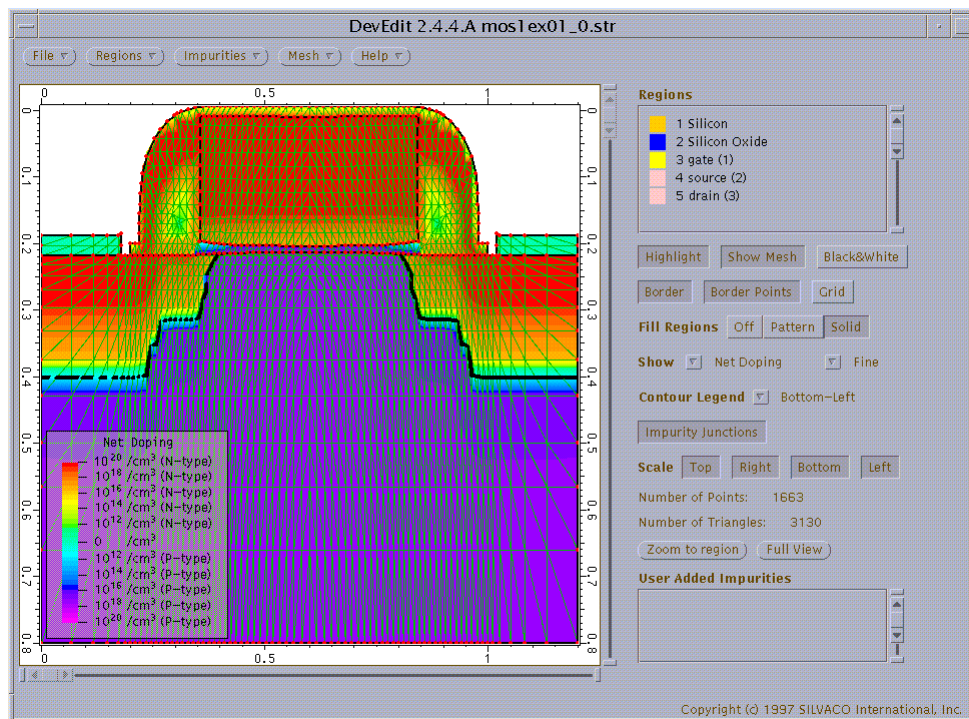


Figure 9-15: Displaying the Net Doping Distribution in DevEdit

Mesh Creation

Boundary Conditioning

There is one important issue that must be considered when creating a new mesh in an existing structure. The first step that you should perform before creating a new mesh is boundary conditioning.

Material boundaries are defined by border points. Necessarily, these border points are also points that define mesh locations. Slight modifications can be made to the structure to minimize the number of border points. This process of eliminating points that are not critical is called boundary conditioning. Examples of such unnecessary points include points along a straight line not contributing to the geometry of the structure, and colinear points (or nearly colinear, such as within one degree).

Right click on **Mesh**, and select **Mesh Parameters**. Both the **Mesh Parameters** and the **Boundary Conditioning** sections are displayed. By default, **Boundary Conditioning** is set to **Automatic**. Note the border points in the structure, denoted as red dots, particularly along the oxide material boundary. If no boundary conditioning were performed, these points would remain and require mesh points at these locations. Click on **Apply**, underneath **Boundary Conditioning**. The mesh is removed, but many border points are removed also.

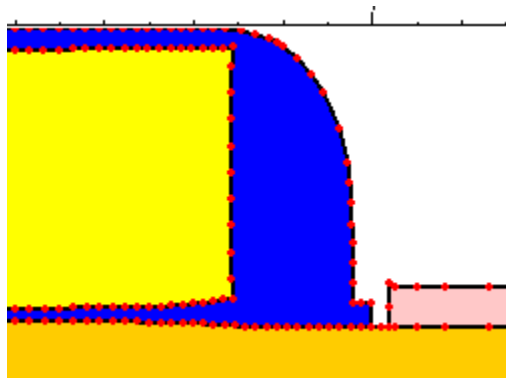


Figure 9-16: Border Points Defining Material Boundaries Before Boundary Conditioning

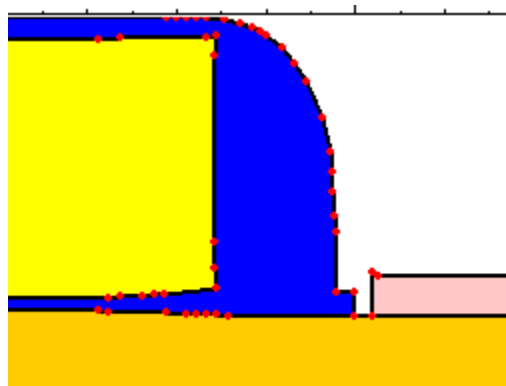


Figure 9-17: Border Points Defining Material Boundaries After Boundary Conditioning

The **Max. Line Slope** is a ratio (greater than one) between the length (x) and height (y) of mesh triangles along a material boundary. If a mesh triangle has a line slope greater than this value, the triangle is subdivided, such that the original material boundary line is divided into two lines, one line horizontal (or vertical), and the other line with an angle set by the maximum line slope.

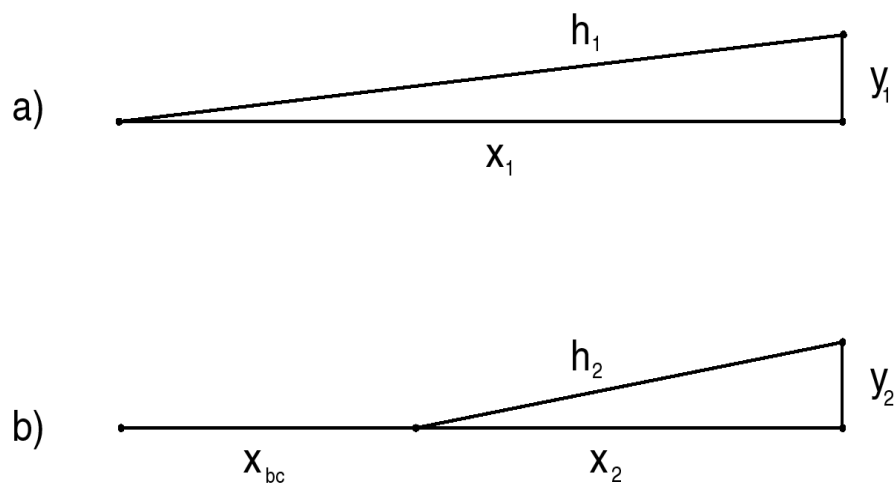


Figure 9-18: Boundary Conditioning with Respect to Maximum Line Slope

Note: The Maximum Triangle Ratio should be larger than the Maximum Line Slope.

The **Rounding Unit** is a distance to which all boundary points are rounded to an even multiple of this value.

The **Line Straightening** value is an angle. If two boundary segments have a joining angle equal to or greater than $(180 - \text{line.straightening})$, the two line segments are combined by removing the joining point.

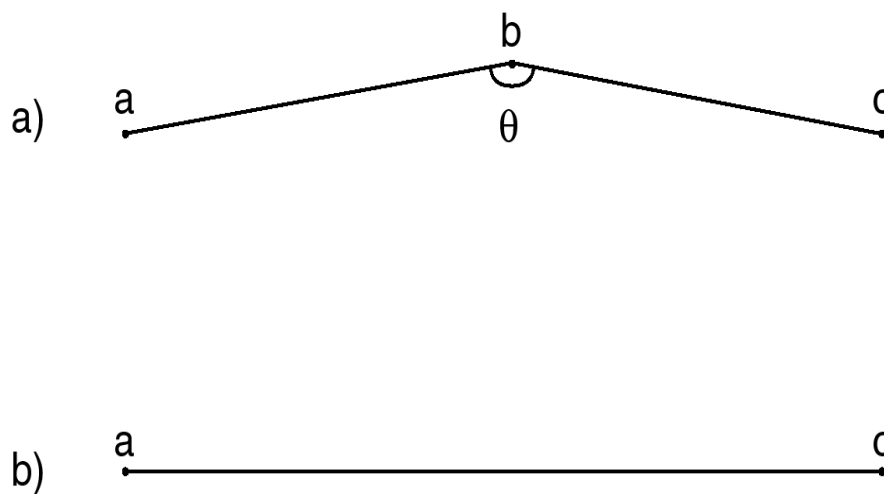


Figure 9-19: Line Straightening

In general, the default settings for boundary conditioning are satisfactory for most structures. After boundary conditioning (i.e., clicking **Apply**), you can proceed to **Refine** on **Quantities and Mesh Constraints**.

Mesh Parameters

Under the Mesh pull-down menu, there is an option of Mesh Parameters, along with Boundary Conditioning. Base Mesh Height and Width may be used to create a fairly uniform underlying mesh. By making these smaller, it may help create more symmetric meshes for some devices. For most devices, setting Mesh Constraints provide the same results, and also allow for more fine tuning. Therefore, these parameters were not used in this example.

Refine on Quantities

This action allows you to refine the mesh on gradients of various quantities, including donors, acceptors, total and net doping, molar composition (if the structure was imported from ATLAS, other quantities including electrical field, and potential, are available). This is a useful tool for refining the grid on areas that require a finer grid, namely, where gradients exist. This tool is very useful, particularly near p-n junctions.

Right click on **Mesh**, then select **Refine on Quantities...**, then right click on the **Add** button, and select **Net Doping**. Then, right click on the **Mesh** pull-down menu, and select **MeshBuild** again (or left click on **Mesh**, which selects **MeshBuild** by default). **MeshBuild** operates by building a mesh with instructions of mesh generation, which have been modified when the **Refine on Quantities** was changed to include gradients in the net doping. The mesh has changed to include a finer mesh where gradients of the net doping exist.

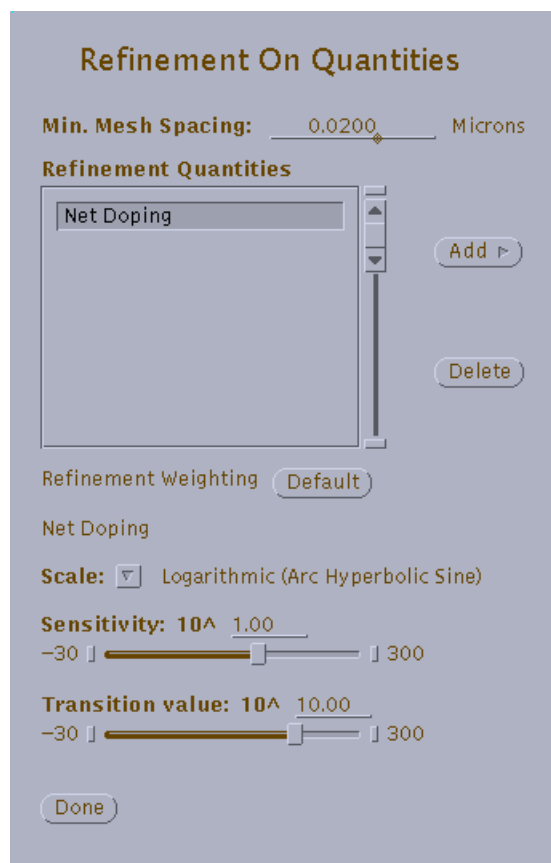


Figure 9-20: Refine on Quantities Menu for Mesh Refinement on Net Doping Gradient

Click on **Done** at the bottom of the panel. This action returns to the main DEVEDIT panel. If the existing panel is not the main panel, click on **Cancel** or **Done** on the exiting panel. DEVEDIT returns you to the previous panel, and ultimately to the main panel. Near the bottom of the main panel the **Number of Points** and **Number of Triangles** are listed. These numbers are useful, in order to gauge the total number of points with the more subjective interpretation of the quality of the grid.

Return to **Refine on Quantities**. The **Scale** default is **Logarithmic**, which is appropriate for doping concentrations. If the value of the quantity varies more than the sensitivity parameter, the mesh is made more dense locally. Accordingly, decreasing the value of the sensitivity increases the number of mesh points. You can change the sensitivity value slightly, then select **MeshBuild** from the **Mesh** pull-down button, and see the effect on the mesh.

The **Transition** value is the minimum value of the quantity that is considered significant. If the **Transition** value is set to 10e10, then gradients of **Net Doping** between 10e9 and 10e8 are not considered when the mesh is created.

Mesh Constraints

The Mesh Constraints section is the principal method for controlling mesh construction. In this section, you can control the maximum triangle ratio, the maximum and minimum height and width, either throughout the device, or selectively in given regions, in material types, or underneath regions or materials.

It is important to avoid obtuse triangles in the semiconductor, however, obtuse triangles in the poly gate, oxide and metal regions are acceptable. A logical method would be to allow obtuse triangles in all regions, but then override the semiconductor regions to only include acute and right triangles.

Begin with selecting **All Regions** in the **Material Types and Regions** box. For **Max. Angle**, either enter or move the slide bar to 150. Next, select **Semiconductor Regions**, then click on the box next to **Max. Angle** and set the value to 90. After re-building the **Mesh**, you can observe that the number of mesh points have decreased. This has been done without adding obtuse triangles to the semiconductor, yet relaxing the mesh in areas that are not of interest.

Further constraints can be imposed in the semiconductor regions. Set the **Max. Height** to 0.1 (all units of distance are microns). Similarly, set the **Max. Width** to 0.1. These values are valid for the mesh creation in all semiconductor regions. As mentioned previously, the mesh in insulators and electrodes are unimportant, so it is acceptable to leave the mesh arbitrarily large in these regions. By clicking on **Meshbuild** again, you will notice that the mesh in the semiconductor now adheres to these criteria.

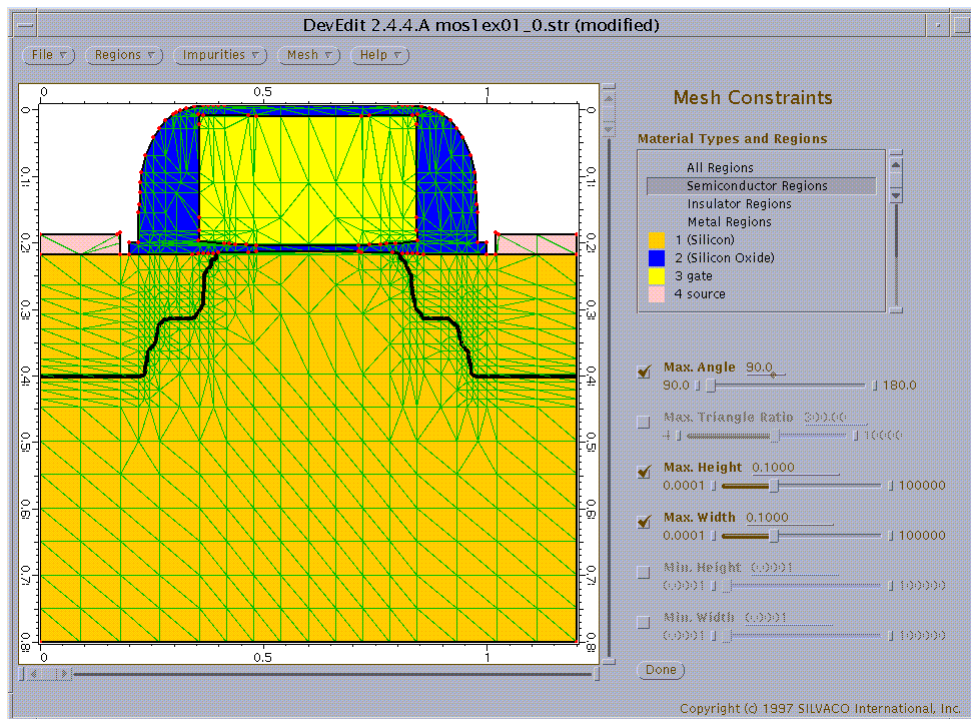


Figure 9-21: Mesh after Refining on Constraints in Semiconductor

Further refinement can be made in more specific areas. Scroll down in the **Material Types and Regions** box. In addition to the general areas (all regions, semiconductors, insulators and metal regions), and the specific regions (in this case, silicon, oxide, gate (poly), source and drain), there are fixed box constraints, under **region constraints** and under **material constraints**. These latter options can be very useful for MOS devices, since the channel area is a critical area for a dense mesh.

Select **Add New Under Region Constraints**. Click on the **Location** button (if not already depressed), adjacent to the **Constraints**. Next to **Under Region**, select (with right mouse button) **gate**. Next to **In Material Type**, select **Semiconductor**. Set **Depth** to 0.1. This procedure has selected the semiconductor underneath the gate, to a depth of 0.1 microns from the silicon surface, as the area for mesh refinement. Next, depress **Constraints**, which displays the same **Mesh Constraints** menu seen previously. Set **Max. Height** to 0.02, and **Max. Width** to 0.04. Click on **Apply**, then click on **MeshBuild**.

At this point, the structure has a mesh that is better suited for device simulation. The original mesh was created primarily for process simulation, which has different requirements, including dense mesh near implant ranges, silicon - oxide interfaces, along diffusion regions, and oxidation areas. For device simulation, the priorities include non-obtuse triangles in the semiconductor, p-n junction boundaries, and areas of high electric field and carrier mobility (such as the channel region in a MOSFET), and do not include oxide shape, electrode shape, and regions far from electrical current.

Note: See S-PISCES chapter of the ATLAS User's Manual for details on typical mesh size required for MOSFETs.

Saving The File

The file can be, and should be, saved in two formats: the structure file and the command file. The structure file is a format used by other Silvaco programs, including TONYPLOT and ATLAS, so it is necessary to save the structure file for continued device simulation. The command file is a list of the instructions used by DEVEDIT to create the mesh (and the structure, if it was made in DEVEDIT, as in Example 1). If you want to make additional changes at a later time, you need to save a command file so that the original set of DEVEDIT instructions can be read. Additionally, running DEVEDIT in batch mode requires the command file.

Structure File

By right clicking on **File**, select **Save as...** and a pop-up menu is displayed. By convention, Silvaco structure files end with the extension `.str`. Choose a name and use this extension (for instance, `example2.str`), then click on the **Save Structure** button. The file can now be used in ATLAS and TONYPLOT.

Command File

In a similar manner, select **Save as...** and enter a name ending with the extension `.de` (for DEVEDIT). It is good practice to use the same filename as the structure file, but ending with the conventional `.de` extension, hence `example2.de` for this case.

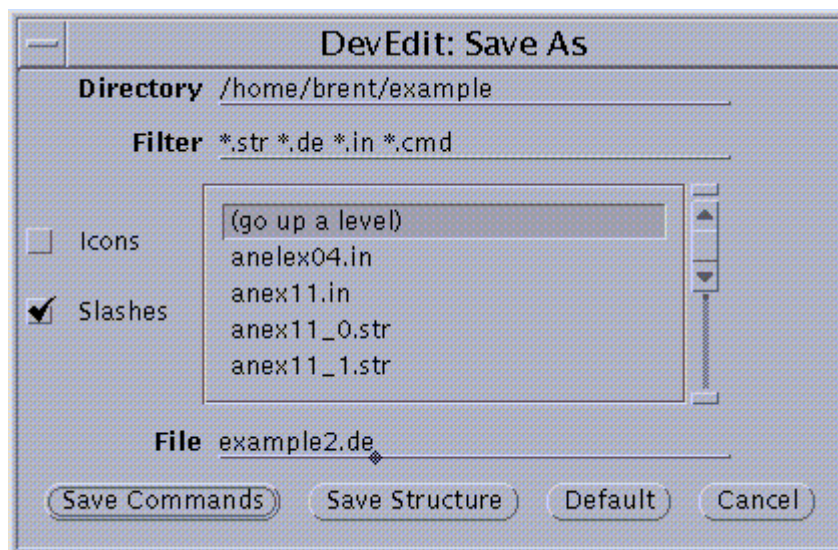


Figure 9-22: DevEdit File Saving Menu

Batch Mode

The goal of DEVEDIT in batch mode for this case is to automate DEVEDIT's mesh creation algorithm, using user defined parameters, without the steps of manual regridding. Therefore, you can make minor changes of the process simulation without requiring you to complete the steps of DEVEDIT in GUI mode, since the criteria used in DEVEDIT (such as mesh constraints in material layers, refined grid along p-n junctions, loose grid deep in the substrat) are general, and would be valid for similar devices of minor process changes. Hence, this purpose for batch mode is different than in Example 1.

In this example, the initial structure was created in ATHENA. Device simulation would be performed in ATLAS. This example demonstrates how to incorporate the DEVEDIT re-mesh syntax within the same input deck, between the ATHENA and ATLAS sections.

You can start any text editor, or use DECKBUILD for editing.

To start DECKBUILD, enter `deckbuild &`, then left click on **File (Open)**, and select a DEVEDIT command file (`example2.de`).

- The first line of the DEVEDIT is similar to DEVEDIT VERSION=2.4.0.R, in which it is recommended that this line become a comment, beginning with #. Then, add as a new first line to read `go devedit` or, if you want to retain the version number, `go devedit simflags="-V 2.4.0.R"` (or whatever the version number of DEVEDIT in use).
- Because the structure was initially created in ATHENA, the DEVEDIT commands to redefine the structure boundaries is redundant and should be deleted. The relevant section begins with **# Set Meshing Parameters**, hence all intermediate lines previous should be removed.
- This truncated DEVEDIT batch file should be moved or copied into the ATHENA/ATLAS input file, beginning after the ATHENA syntax, and before the ATLAS syntax (See figure batchDevEdit.pcx. The last line of the ATHENA section is `tonyplot moslex01_0.str -set moslex01_0.set`, and ATLAS begins with, `go atlas` at the bottom of the DECKBUILD main window).

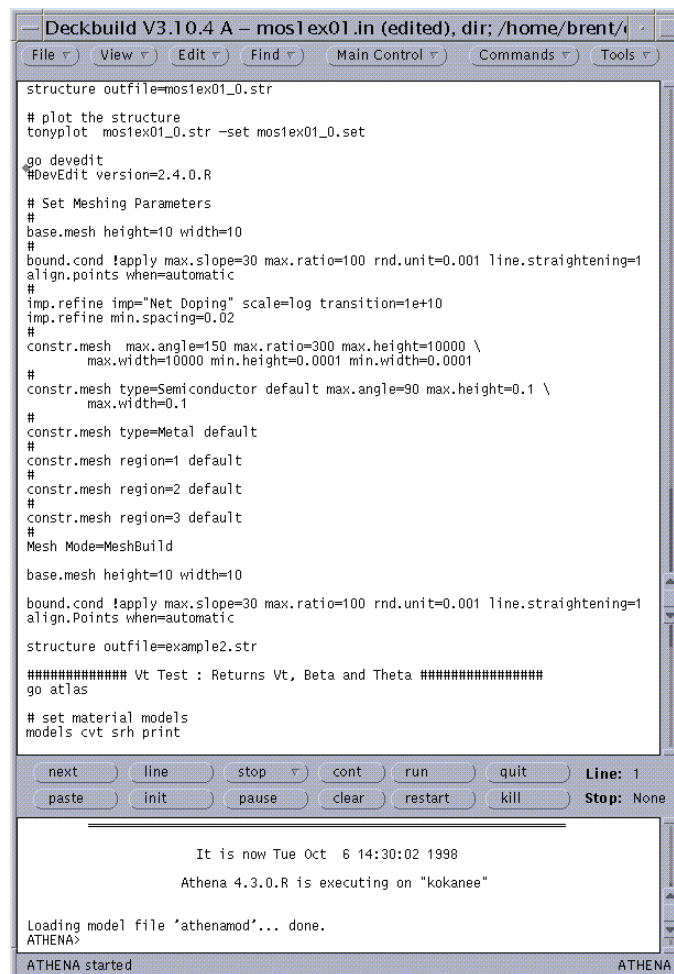


Figure 9-23:Running DevEdit in Batch Mode within DeckBuild after ATHENA Process Simulation

- We also recommend that you add the line `structure outFile=example2.str` at the end of the DEVEDIT section of the input file to explicitly save the structure in a structure file. Any filename can be used with the convention of the `.str` extension used for structure files.

With this procedure, you can run the entire deck and incorporate the mesh re-creation from DEVEDIT. Any minor changes in the ATHENA process simulation file does not require changes to the DEVEDIT command file.

An alternative method of using DEVEDIT in batch mode is to use the command file as a distinct file, and not incorporate it within ATHENA-ATLAS input file(s). As mentioned above, you must edit the file to remove the structure definition (up to the `# Set Meshing Parameters` line). However, DEVEDIT must initialize from the original structure. This can be accomplished by including an **initialize** command immediately after commencing DEVEDIT, hence:

```
go devedit init infile=mos1ex01_0.str
```

then continue with the command file. The ATLAS file would then begin with

```
go atlas mesh infile=example2.str
```

or whatever name of the structure file the user-defined at the end of the DEVEDIT command file.

9.4.4: Advanced Features

3D Structures

Treatment of three dimensional structures in DEVEDIT3D is much the same as the two dimensional version, with the exception of the Z plane. The handling of region addition and modification, electrodes, impurities, mesh parameters and constraints, and boundary conditioning, are all done similarly in 3D as 2D. Any user wishing to use DEVEDIT3D should first read either or both of these examples.

DEVEDIT3D creates a prismatic based 3D solid consisting of several 2D planes. Each region is initially defined throughout all Z planes, then the region must be defined for beginning and ending Z planes.

One capability of DEVEDIT3D is to extend an existing 2D structure (created in ATHENA) into three dimensions. Such an example is discussed below.

Start DEVEDIT3D by entering `devedit3d &` in a UNIX terminal window. In a similar fashion to Example 2 (above), start DECKBUILD and load example `mos2ex04` into DECKBUILD. In DEVEDIT, load in `mos2ex04_0.str`, by right clicking on **File**, then **Load...**, a **DevEdit: Load** window opens; select the file `mos2ex04_0.str`, and enter an **End Z:** value of 1.1, then click on **Load File**.

You can modify the polysilicon gate in the same manner as the 2D version by selecting the region and select **Regions**→**Modify Region...**. The X and Y points are maintained, but the **Start Z:** and **End Z:** planes should be changed to 0.3 and 0.8 respectively. Click on **Apply**.

In order to add aluminum electrodes, select **Regions, Add region...**, and specify the X and Y coordinates (either by mouse or by keyboard) of (0.15, -0.15), (0.15, 0.05), (0.6, 0.05), and (0.6, -0.15). The **Start Z:** and **End Z:** should be set to 0.0 and 0.1. Toggle the **electrode** button and select **drain**, then click on **Apply**. A similar set of X and Y coordinates should be used to define a source contact, but from `Z=1.0` to `Z=1.1`.

The remaining difference between the 2D and 3D versions of DEVEDIT is the Z-plane mesh. Right click on **MESH**, then select **Z planes**. Mesh planes in the Z plane are automatically inserted at material boundaries. These mesh planes are listed in **boldface**, and cannot be modified or deleted. Additional mesh planes are included, and are listed in *italics*, which can be edited.

You have three methods of controlling the Z plane mesh:

- Max. Plane Spacing, which limits the distance in the Z plane of the mesh.
- Max. Spacing Ratio, which is the ratio of the spacing from one neighboring mesh plane to the next,
- or
- Selectively add, modify, and delete specific mesh planes by entering the location and spacing of the plane.

DEVEDIT3D, however, overrides modifications you have entered to maintain the **Max. Plane Spacing** or **Max. Spacing Ratio** constraints.

There are no 3D graphics in DEVEDIT3D. To view the 3D structure after meshing, a structure file should be saved. This file can be loaded and displayed in TONYPLOT3D.

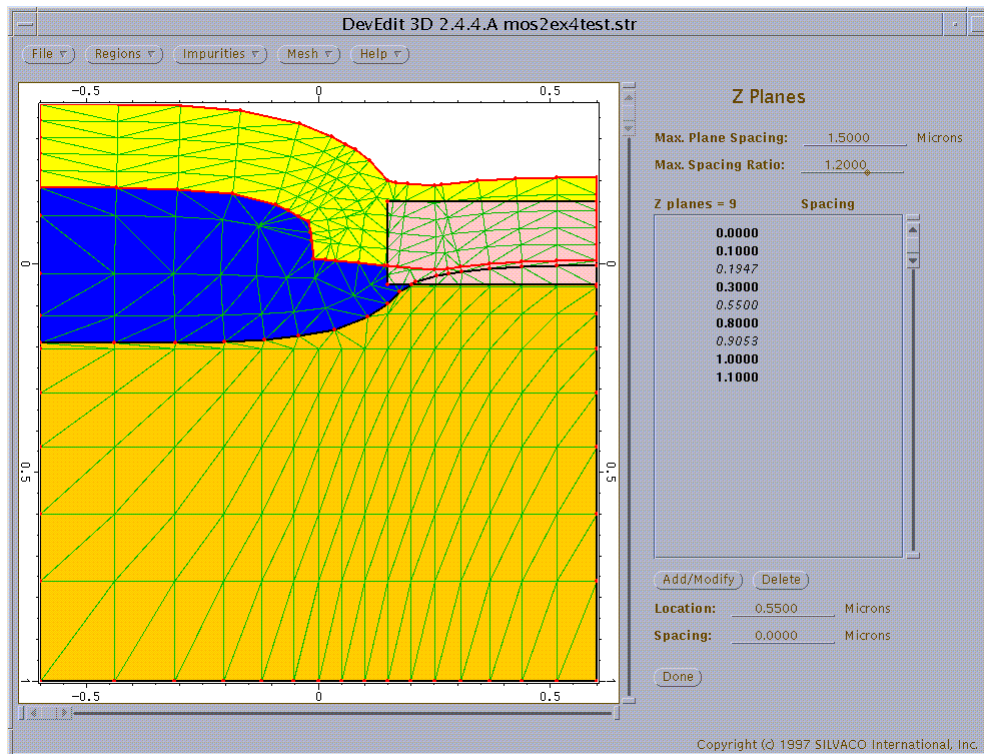


Figure 9-24: Three-dimensional Mesh Creation in Z-plane

Combining Two ATHENA Structures into a Single Device

Since CPU time required to run process simulation is super-linear with the number of grid points, there can be significant CPU time savings by splitting a large simulation into sections. These sections can be joined together at the end of the process simulation using DEVEDIT.

The **JOIN** function combines the device currently loaded in DEVEDIT with another saved structure into one device. There are controls for aligning the edges of the materials or regions.

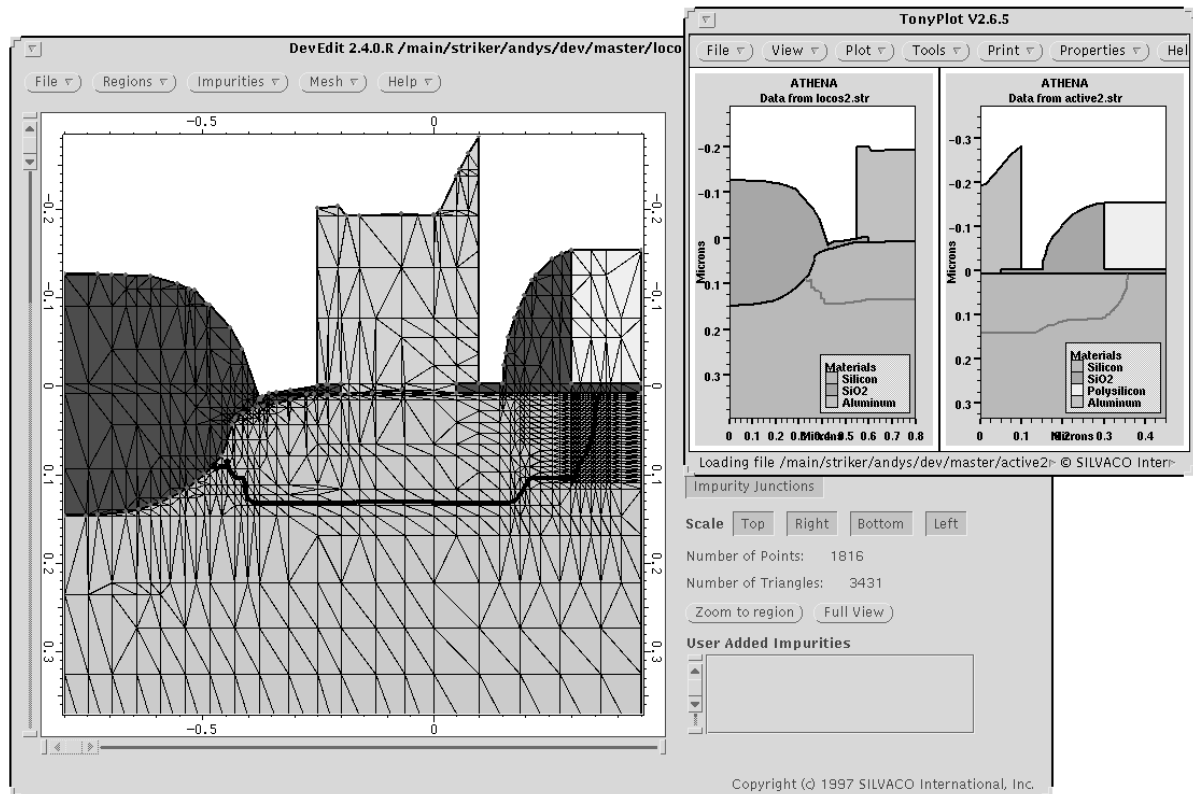


Figure 9-25: Device Structure Joined in DevEdit

Stretch and Cut

DEVEDIT has three commands, **STRETCH**, **SQUEEZE**, and **CUT**, to modify the structure and to increase or decrease height or width. Both commands are available in both the GUI mode and batch mode of DEVEDIT. These commands are under Regions in the GUI mode of DEVEDIT.

STRETCH and **SQUEEZE** functions allow a line or region to be expanded or contracted along the vertical or horizontal directions. For rounded shapes, the curvature is altered, but left intact. Stretch does not work as a simple **add** function; otherwise it causes a discontinuity in the curvature.

With an existing structure in DEVEDIT, select **Stretch/Squeeze...** under **Regions**. The Direction buttons allow for stretching either horizontally (left or right) and vertically (up or down). The mouse or keyboard can be used for **Start** and **End** locations. The **Old Size** indicates the existing size of the line or region selected. The **New Size** allows you to adjust the region size to either smaller or larger than the existing size.

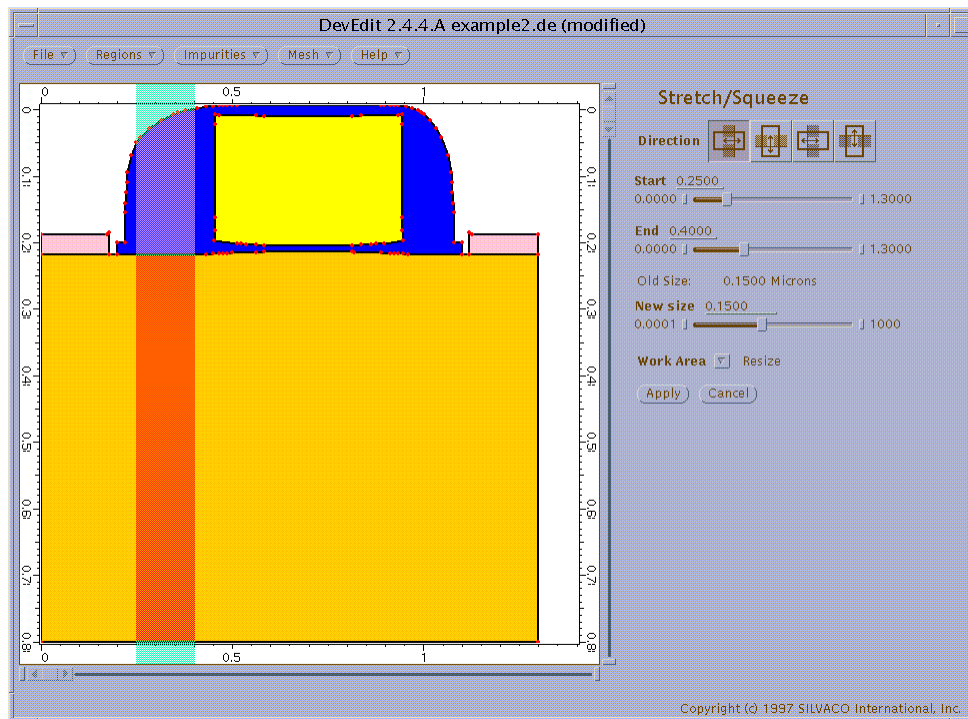


Figure 9-26: Stretching an Existing Structure

The **Cut...** function works in a similar manner. You can select the area using the mouse or keyboard, and delete the area.

Circular Devices

DEVEDIT also allows for creation of circular objects. Both circles and arcs can be created. This feature is unique to DEVEDIT within the Silvaco VWF suite of tools - neither ATHENA nor ATLAS can easily create a circular region.

Select **Add Region**, under the **Region** pop-up menu. Right click on **Move/Add Point**, and select **New Circle/Arc**. The right panel then displays **Create Circle or Sector**. The first **Mouse mode** setting is **Set Center...** Click on a point in the work area to select the center of the circle. After selecting a center point for the circle, the **Mouse mode** changes to **Set Radius and Start Angle...** You can select a complete, closed circle, or restrict the region to a portion of a circle. Once chosen, the **Mouse mode** changes to **Set End Angle...** The **Points around circle** option allows you to determine the number of points of the circle. DEVEDIT is still approximating a circular region with straight interconnecting lines. But you can choose between 30 degree segments (12 points) to one degree segments (360 points); the latter approximating a circle quite well.

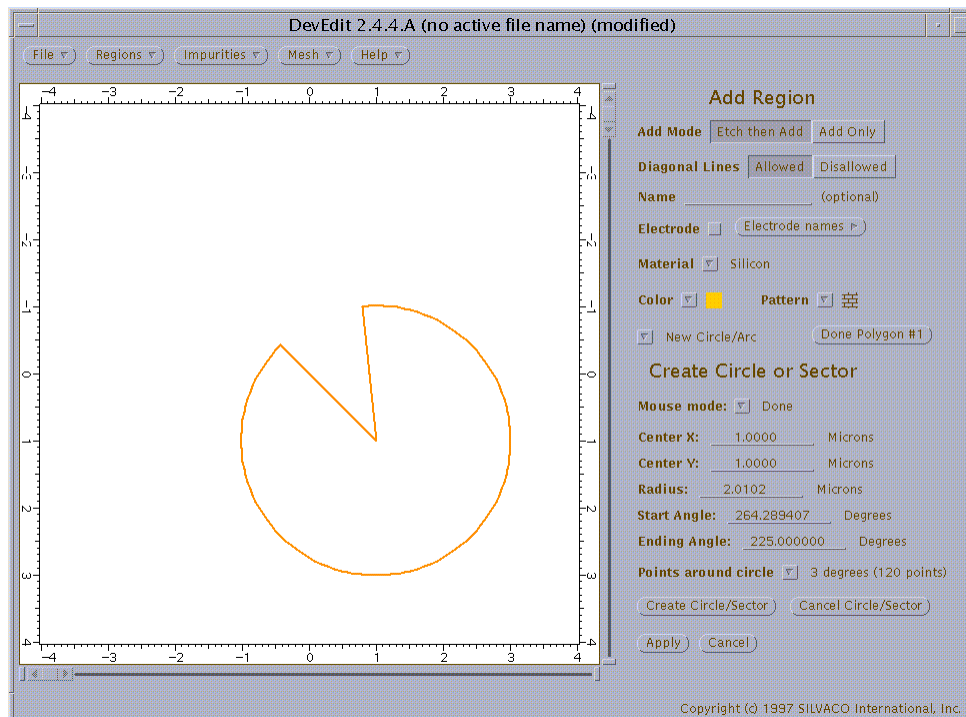


Figure 9-27: Creating a Circular Region

Summary

The goal of creating a mesh in DEVEDIT is to accommodate competing interests of numerical efficiency and accuracy in device simulation. DEVEDIT can be used to create a device structure in a graphics user interface mode (as in Example 1), or create a new mesh in an existing structure (as in Example 2). Parameters that you defined control the advanced mesh algorithm within DEVEDIT. Advanced features in DEVEDIT include 3D structure creation and prismatic mesh generation, combining two structures into a single device, stretching and cutting, and circular device creation.

Beginning users can find examples in DECKBUILD EXAMPLES, of DEVEDIT in batch mode. Questions are also be entertained by Silvaco's Applications Engineers.

9.5: STRUCTURE EDITING

9.5.1: Overview

This section describes the general editing functions that are available in DEVEDIT. DEVEDIT stores a history of all events that have taken place under DEVEDIT control. These events are stored internally and can be written to a **Command** file (see Section 9.10.6: “Saving the Silvaco Standard Structure File”).

Zooming

The Zoom function is completed by holding the left mouse button down and dragging over the desired zoom area. In the event of needing to zoom in half way through object definition when the mouse button can not be held down, holding the Shift key down overrides any editing actions and zooms in. Zooming out is accomplished by double clicking. This action zooms out by a factor of four times.

Panning

Panning over a structure is accomplished by moving the scroll bars situated initially at the top right and bottom left of the editing screen. Holding the left mouse button down while selecting either end of the scroll button moves the displayed structure to the left and right and up or down. A single click on the scroll bar end stops and pans all the way across the displayed structure. Panning can also be accomplished by holding the middle mouse button down on the inside the main edit window, while dragging the structure vertically or laterally across the screen.

Editing Summary

The capability of DEVEDIT to define structures on the screen is explained in the next three sections (Material, Doping, and Meshing). A region is generally defined as a piece of material structure, for example, a gate, an oxide, an aluminum contact, or a poly layer. Adding a material region to the DEVEDIT structure is accomplished by following five simple steps.

1. Select the required resolution.
2. Select the **Add Region** mode.
3. Select the material, its color and doping concentration
4. Draw the region on the screen.
5. Click on the **Apply** button.

9.5.2: Selecting The Resolution

When drawing the material region, **DevEdit** snaps boundary point locations of the newly defined region to the resolution defined by the drawing grid. A drawing grid displayed as tick marks on the **Main** panel can be overlaid by clicking on the **Grid** button. If a very thin layer is required, the tick marks may not allow this resolution. Increasing the resolution can be accomplished by zooming into the region of interest. It may also be useful to split the screen allowing the use of more than one drawing resolution. Thus, the resolution of the screen should be chosen before adding a material region.

Note: Zooming in can be accomplished mid-way through a region definition by pressing the Shift key down. The Shift key overrides the current action so the left mouse button will always control a zoom in function while the Shift key is held down. Region definition can continue once the Shift key is released.

9.6: EDITING REGIONS

9.6.1: Adding a Region

The **Add Region** mode is invoked by selecting **Add** from the **Regions** menu. Having done this, the right side of the DEVEDIT **Base Window** changes to show the **Add Region** control panel. The **Add** mode can be selected at the top of the **Add Region** panel. Two options exist and determine the way regions are added to an existing structure. The **Add Only** option adds a region without recessing into an existing region. In this mode the existing regions take precedence over the new region being added. The **Etch then Add** option recesses a new region into an existing region. The new region being added takes precedence over the existing regions.

Note: A section can be etched away or deleted by creating a new region using the **Etch then Add** mode, and deleting the whole region after defining it.

9.6.2: Selecting Region Material

A region's material should be selected before, or during, the process of region definition. Although Silicon is the default material, other materials can be selected from the **Material** menu on the **Add Region** panel. A large number of materials are currently available on the **Material** menu, and more materials can be provided by contacting the local Silvaco representative.

Each material is assigned a different color by default. This can be changed, if two regions have to be distinguished, by selecting a color from the **Color** menu on the **Add Region** panel, as shown in Figure 9-28. Colors are selected purely for DEVEDIT visualization. The color information is not saved into the final structure file and so it is not applicable for later applications, i.e., TONYPLOT.

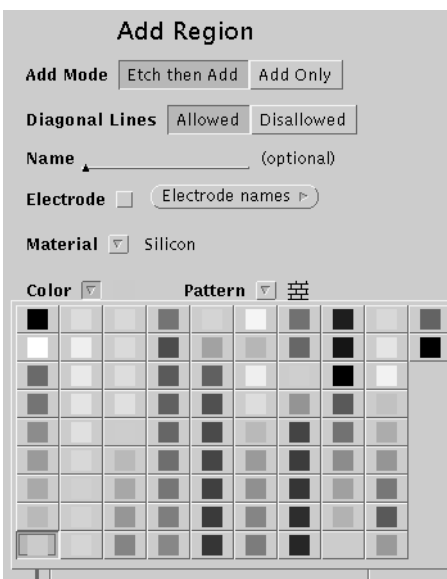


Figure 9-28:Color Menu

9.7: DRAWING REGIONS

9.7.1: Overview

A region is normally defined by a polygon which denotes its outer boundary. This boundary can be drawn by clicking the left mouse button on the first point of the polygon. (The point is placed when the left mouse button is released, not when it is pressed. This is to match the feel when points are dragged, and is described later). This starts a new polygon. After placing the first point, a line is rubber-banded to the mouse until the second point is placed. After that, two lines are rubber-banded; one from the previous point to the mouse, the other from the mouse to the first point creating a polygon. Points are added until the **Done Polygon** button or the right mouse button is clicked. (When the **Done Polygon** button is displayed, you can click on it with the right mouse button in the drawing window).

For example, to define a rectangular region; click the left mouse button four times in the screen at the vertices required to make up the rectangle, then the right mouse button to finish the polygon. Selections from the right side control area can be made at any time. An example of this operation is shown in Figure 9-29, where two regions have been defined, with a third region partially defined.

After clicking on **Apply**, each region added to the screen is listed as a material and as a number on the scrolling list in the region screen. Regions can be selected by clicking left on an entry in the scrolling list. A selected region is surrounded by a red outline in the main edit window.

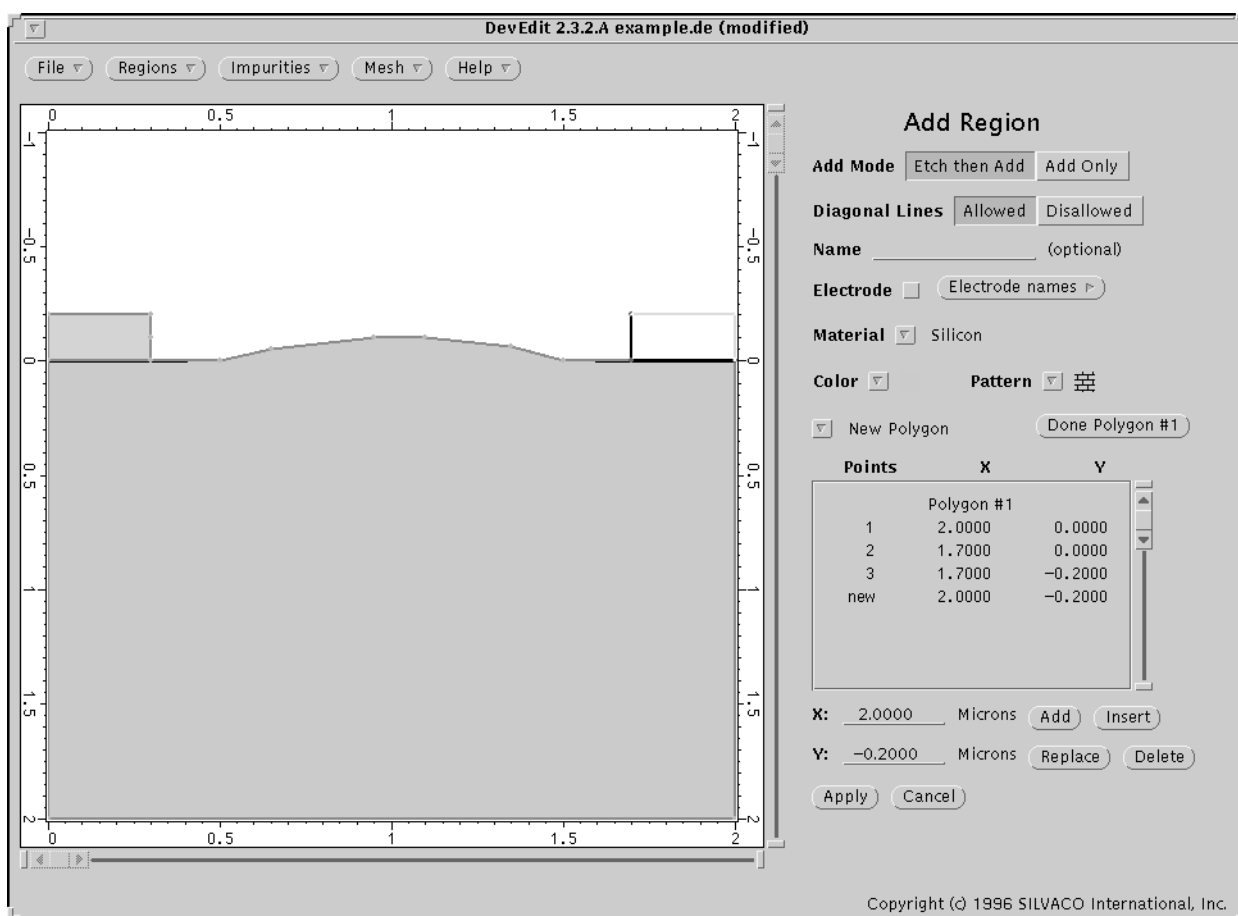


Figure 9-29: Drawing a Region

Regions can also be made from multiple polygons. For example, an oxide region can completely surround a polysilicon region, thereby having two polygons (an outer edge and an inner edge). Normally, this would be done by first adding the oxide as one polygon. Using the **Etch then Add** mode, the polysilicon region can be added to change the oxide region in the process. You can add a second polygon, after the first polygon has been completed, by selecting **New Polygon** or by clicking on the left button at the starting location for this new polygon. Use the **New Polygon** option, if you wish to start the new polygon near an existing point or line. Otherwise, it is considered a modification operation and does NOT start a new polygon.

Alternatively, a region may contain no area but only describe an interface line. This is sometimes used for electrodes, to limit the mesh size or show a contact beginning at a simulation limit. To draw this type of region, before adding any points select **New Line** instead of **New Polygon**.

A polygon can also be created by selecting **New Circle/Arc**. This forms a regular polygon, or a piece of a regular polygon, after selecting various parameters. These parameters include the center of the polygon/circle, a radius, a start and ending angle, and the number of faces (angle between points).

9.7.2: Setting Base Impurity

A region can be made of a material with a constant base doping. This property can be defined by selecting **Set Base Impurities** (this completes any shape editing currently being performed, if that is impossible. You can not change the base impurities until the region has a consistent shape while adding or modifying a region). The impurity types are separated into four categories. This is a list of the supported impurities:

1. Silicon Donors
 - Antimony
 - Arsenic
 - Phosphorus
2. Silicon Acceptors
 - Aluminum
 - Boron
 - Gallium
 - Indium
3. Generic Donors/Acceptors
 - Donors
 - Acceptors
4. Composition Fractions
 - Comp. Fraction X
 - Comp. Fraction Y
5. Net Doping (directly)

Some of these values can also be set, using **Resistivity** tables by clicking on the **Set Si Resistivity**.

9.7.3: Deleting Regions

A region can be deleted by first selecting the region to be deleted and selecting the **Delete** option under the **Regions** menu. To select a region, select the region listed in the scrolling list of regions on the main panel. A selected region is surrounded with a red colored highlighted border in the main screen. Adding a region can be used to etch a region out of an existing structure.

9.7.4: Modify Regions

To modify a region, select the region on the main panel in the region list, then pull down the **Regions** menu and choose **Modify**. Now the region parameters can be changed as in the add region.

9.7.5: Deleting Boundary Points

Boundary points on a region's boundary can be deleted at any time. To delete a single point or a number of points, select the **Delete Points** option. The cursor appears as a scull and cross bones. Hold the left mouse down and drag a box over the points to be deleted. Upon releasing the mouse button, the points are deleted and the region modified. This may be important in minimizing the number of unnecessary points leading to large numbers of triangles.

9.8: DOPING DEFINITION

9.8.1: Overview

Doping can be defined analytically with DEVEDIT. Doping can be graphically defined by indicating an area or line. This line or box is given attributes that describe in detail the doping to be added to the structure.

9.8.2: Defining an Impurity Source Line

Doping can be defined about a line source drawn on the device. The line source can be vertical or horizontal to the main axis. A doping line is defined by choosing the **Add** option from the **Impurities** menu. The area to the right of the base window appears, as shown in Figure 9-30. The **Line Mode** switch in the **Impurities** control window should be switched to **Line**. A line source has a constant doping concentration along that line segment and the concentration can roll off in magnitude in any perpendicular direction from the line segment. The line is defined on the screen by clicking the left mouse button at the start of the line and clicking again at the end of the line. Once drawn the line can be adjusted by clicking a revised point at either end of the visible line. The end of the line to be modified is determined at about the half way point along the line. Thus, modifying an existing line is very simple so that placement can be very accurate. The position of a line is fixed only when the apply button is pressed on the main doping control screen.

Figure 9-30: Impurity Panel

9.8.3: Defining an Impurity Source Box

A doping distribution can be defined about a box drawn on the device. The box source can be vertical or horizontal to the main axis. A doping source box is defined by first selecting the **Add** option under the impurities menu button. The control screen to the right changes at this point. The **Draw Mode** switch in the **Impurities** control window should be switched to **Box**. A **Box** source implies a constant doping concentration within the box and the concentration can roll off in magnitude in any perpendicular direction to the box. The box is defined on the screen by holding down the left mouse button at a box corner and dragging the cursor over the screen. Once drawn the box can be adjusted by moving the top

right and the bottom left corners of a box. A corner can be moved by holding down the left hand mouse button at the corner to be moved while dragging the mouse. Thus modifying an existing box is simple so that placement can be very accurate. A box can also be placed by defining the coordinate points of the boxes sides. These fields can be defined by filling in the Start X, Start Y, End X, and End Y text field entries. This feature is often useful for more exact placement. The position of a box is only fixed when the **Apply** button is clicked on the main doping control screen.

9.8.4: Doping Source Attributes

Doping can be defined to roll off in perpendicular directions to the source boundary (either a line or a box). These attributes are simple analytical expressions that have traditionally been used in the description of semiconductor devices. These functions can be listed as follows:

Gaussian

- Error Function
- Linear
- Logarithmic
- Exponential Constant
- Step function

Each of the functions refer to the way the doping distribution decays with distance from the edge of the doping source (box or line).

9.8.5: Deleting Source Objects

A doping source object can be deleted by selecting the object from the list on the top level menu, then selecting the **Delete** option from the **Impurities** menu. A selected object is surrounded by a red line on the screen.

9.8.6: 3D Doping

Three dimensional doping objects can be defined in the case of a 3-D structure. DEVEDIT has to be invoked in the 3-D mode for this case to apply by entering:

```
devedit3d &
```

X, Y and Z parameters should be specified, in the case of a 3-D doping objects being added to the structure. As in the 2-D case, this action is controlled from the **Add Impurities** control panel, as shown in Figure 9-32

9.9: MESHBUILD MESHING

9.9.1: Overview

A structure can be meshed by selecting the **MeshBuild** option under the **mesh** button. The **MeshBuild** code originated from ETH Zurich and has been enhanced for use inside the DEVEDIT environment. **Meshbuild** starts with a mesh adapted to the defined boundary. Efficient meshes require that the boundary is conditioned before meshing. This means the boundaries are altered slightly to accommodate the meshing algorithm. **Meshbuild** allows the selection of arbitrary boxes in space for either manual mesh refinement or manual mesh relaxation. **Meshbuild** can be passed a solution quantity on the mesh (i.e., boron concentration), whereby it automatically refines the mesh based upon the gradient of the quantity. A structure can be meshed by selecting the **Meshbuild** menu option under the **Mesh** menu.

Note: Boundary conditioning is strongly advised just *before* meshing any structure, although it is not essential.

During the mesh routine, an ETH (Zurich) Meshbuild Acknowledgment screen appears. The meshing can be canceled at any time with the **Cancel** button on this screen.

9.9.2: Boundary Conditioning

The Boundary conditioning menu option can be found under the **Mesh** menu **Mesh Parameters...** item. When **Mesh Parameters...** is chosen, the **Boundary Conditioning** control panel appears, as shown in Figure 9-31.

Boundary conditioning should be used before any mesh is created. If a structure is modified the boundaries should be Conditioned before remaking the mesh.

Mesh Parameters

Base Mesh Height: 0.2500 Microns

Base Mesh Width: 0.2500 Microns

Max. Triangle Ratio: 100.0000

Boundary Conditioning

Off Manual Automatic

Max. Line Slope: 30

Rounding Unit: 0.0010 Microns

Line Straightening: 1 Degree(s)

Try to Align Points: No Align

Apply Cancel Default Restore

Figure 9-31: Mesh Parameter Panel

9.9.3: Limitations

A few basic limitations of **Meshbuild** should be realized before starting to use the mesh generator. **Meshbuild** creates a mesh with few obtuse triangles. This improves convergence during subsequent solutions. In order to maintain this criteria, **Meshbuild** adds a large number of triangles in or around border points. For this reason, the number of border points should be minimized. Slight modifications can be made to the structure to minimize the number of boundary points. This concept is called **Boundary Conditioning**. The **Refinement Limits** option under the **Mesh** button sets up the boundary conditioning control panel. Here a number of approximations can be made to improve the meshability of the structure. Most structures can be conditioned by clicking on the **Apply** button; this should be the first option. **Boundary Conditioning** attempts to get rid of points that are not critical.

Boundary Conditioning falls under the following categories:

- Any points on a straight line not contributing to the geometry of the structure are eliminated.
- A geometrical rounding error can be supplied with a default limit of 0.001 micron. Care should be taken in view of this critical layer thicknesses when using this option. (i.e., gate oxides). This option is controlled with the **rounding unit** text field on the boundary conditioning control screen.
- A line straightening algorithm is available. The option straightens a line if it bends by only a small amount defined as an angle in the **line straightening** text field.
- **Meshbuild** essentially refines a basic, coarse, tensor product mesh. The spacing of this base mesh can be controlled. Two options for the X and the Y directions are available to do this on the boundary conditioning control screen.

9.9.4: Mesh Constraints

Mesh constraints allow you to require specifications be met in the mesh. While mesh parameters are guidelines for the meshing algorithm, mesh constraints are requirements that must be met (see **Maximum Mesh Angle** for exception). There are two major parts in defining mesh constraints; the effected area and the constraints being opposed on that area. The types of effected areas are described in the next few paragraphs and the types of constraints that can be applied.

Constraint areas are either region based or rectangle based. Region based constraint areas are set up in a hierarchy, where the most specific constraint area overrides the less specific areas that apply. Rectangle areas, as you may have guessed, are rectangles. They are specified by absolute coordinate or in relationship to regions (as under metal regions). When two rectangles overlap, the constraint applied to the overlapping area is the most restrictive constraint. When region constrains and rectangle constraints overlap, the most restrictive constraint is also used.

In region based constraints, each region has its own set of mesh constraints. If a specific constraint is not set for a region, the value from the mesh constraint specified for that region's material type is used. If it is not set, the value from the **All Regions** mesh constraint is used. This allows setting constraints for all regions, a specific material type or a specific region.

The scope list allows specification of which set of mesh constraints are being set. The check box to the left of each value must be turned on if the value is to be changed. For **All Regions**, the check boxes are always set because those values are the default values for all other scopes. For material type scopes (i.e., **Semiconductor Regions**, **Insulator Regions**, and **Metal Regions**), turning on the check box sets the default value for regions made of that material type, overriding the **All Regions** value for those regions. If the check box is set for a particular region, the value to the right of the check box applies only to that region.

Note: When the check box is turned off, the value displayed is the current default value for the current scope.

Finally, we get to the constraints that can be imposed.

max.angle - Maximum angle of a triangle in the specified region or scope. In certain cases, limitations in the mesh build algorithm require a few triangles to be obtuse, no matter what the maximum angle is set to. This keeps mesh build from creating an infinite number of triangles. Work is currently ongoing at Silvaco to resolve this restriction. (If the angle is less than 180 degrees the maximum connectivity of a node is 12).

max.triangle.ratio - Maximum ratio of a triangle's height and width.

max.height - Maximum height of a triangle.

max.width - Maximum width of a triangle.

min.height - Triangles with height less than this value, are not refined in the Y direction during impurity refinement. Some triangles may be shorter than this value to allow for geometry

min.width - Triangles with width less than this value are not refined in the X direction during impurity refinement. Some triangles may be narrower than this value to allow for geometry.

9.9.5: Adaptive Meshing

Adaptive meshing is an efficient way of adding grid points in areas of interest semi-automatically without having to add too many points. The purpose is to search for steep gradients over a solution and to add grid points locally to these regions. A mesh can be adapted after creating a basic boundary compliant mesh. A mesh can be adapted to the gradient of any impurity quantity on the mesh. To generate an adapted mesh, select the **Refine on Impurities** option from the **Mesh** menu.

9.9.6: Refinement

Refinement on impurity concentration gradient control shows the control screen for impurity defined mesh refinement. The **Minimum Mesh Spacing** control refers to the minimum size of a mesh element after adapting. A mesh is not be refined down below this entered value even in the case of large concentration gradients. Available impurities can be selected from the **Add Menu** option. All impurities present in the current structure are available to mesh with, i.e., in the case of a SPICES2 solution, a large number of impurities are available including Electron Temp, doping conc, potential etc. Any number of impurities can be selected from this menu for addition to the scrolling list of impurities.

Refinement on impurity concentration gradient control show three impurities: boron, arsenic and phosphorus. The concentration gradients of these selected impurities is used to adapt the mesh. Each impurity has its selected weight, i.e., boron = 0.85. Each selected impurity can be weighted by selecting the impurity on the scrolling list and sliding the weighting factor slider to the required weight. The smaller the number, the finer the grid. The weighting factor refers to the natural log increase of a quantity that is accepted without halving the mesh spacing. For example, if the weight for phosphorus was set to 3, the grid would only half its size when the concentration was found to be more than $(\exp 3) = 20.03$ X difference between two adjacent points. By default, the weight is 1, so each time a value is multiplied by 2.72 across two grid points, it is earmarked for adapting. Selected impurities can be deleted from this list by clicking on the listed impurity and clicking on the **Delete** button. Impurities have to be present in the structure before refinement can take place. Once weights have been setup, click on the **Apply** button. To start the meshing procedure, select the **Meshbuild** option under the **Mesh** button. **Boundary conditioning** is advised before any meshing procedure. **Boundary conditioning** can be applied either before or after setting up the adaptive meshing settings.

9.9.7: Manually Refining The Mesh

Any mesh can be refined manually by selecting the menu options. Users can refine a selection box in both directions. Alternatively, refinement can be restricted to either the X or Y direction by selecting either of the options: **Mesh - Refine Box - Refine X** OR **Mesh - Refine Box - Refine Y**. A selection box is specified by holding down the left mouse button, and dragging over the area to be refined. Each element in the box is reduced in size by a factor of two per direction. The mesh inside the selection box can also be relaxed in a similar way, by selecting the **Unrefine** menu option. **Meshbuild** can cause mesh outside of the defined box to be effected also.

9.9.8: Manually Relaxing A Mesh

A mesh that has been refined manually or refined adaptively on an impurity gradient can be relaxed. A mesh that was generated purely from a set of boundary points cannot be relaxed. To relax a mesh, select the menu options: **Mesh - Refine Box - Unrefine**. Define a box by dragging the mouse over an area with the left mouse button held down. The mesh is refined in the area of the box. **Meshbuild** may cause mesh outside of the defined box to be effected also. The unrefine attempts to double the size of each element inside the defined bounding box.

9.9.9: Tensor Product Mesh

A tensor product mesh is one that runs from top to bottom and from left to right at every point on the mesh. It is highly inefficient but produces straightforward matrix conditioning. After a structure has been defined it can be meshed using a tensor product algorithm.

DEVEDIT supports a modified tensor product mesh using the following algorithm. All boundary points are used to create a true tensor product mesh, however, since there can be new points created during that process, the **MeshBuild** algorithm is used to resolve these new points. This also allows the resulting mesh to be refined using the **Refine Box** commands. To create a modified tensor product mesh, select **Tensor Product** under the **Mesh** menu button.

9.9.10: Work Area Resizing

The work area can be resized by selecting the **Resize Work Area...** menu option from the **Regions** menu button. New coordinates can be added to the **Resize** work area control panel. Click on **Apply** when done.

9.9.11: Defining 3D Structures

A prismatic based 3-D solid can be specified in terms of a number of 2-D planes. This is the approach DEVEDIT uses. A super set of all regions are defined, covering all potential, Z-plane. Then, individual regions are assigned a start and end Z-plane. As an example, Imagine a MOSFET conducting into the plane of the screen. A poly gate runs from left to right across the screen. Field isolation regions appear on the left and right hand side of the screen also. You need to specify the values of the poly region in the Z direction. These two values, start and stop, define the gate length. Similarly, all contact metal regions can be defined with a start and stop value in the Z direction. As a region is defined, the Start Z and Stop Z fields should be specified. These values indicate the depth, into the screen, that the region boundaries apply. Thus a 3-D structure can be defined from a single elevation.

9.10: CREATING A NEW MESH

9.10.1: Setting The Mesh Controls

To create a mesh, the first step is to set the meshing controls. This is accomplished by setting the **Meshing Parameters**. To do this, pull down the **Mesh** menu and choose the **Mesh Parameters...** item. A new control panel appears on the right side of the DEVEDIT base window.

9.10.2: Base Mesh Parameters

Base Mesh Parameters are a guideline to create a mesh that fit the structure. Set the width and height to 0.2 Microns. **Boundary Conditioning** controls simplification of the input structure. When the structure is first read in, all the original boundary points appear as red dots. Boundary conditioning moves these points slightly, or eliminates them if the points are not needed. At this point, click on **Apply**. It may be noticed that some of the red dots have disappeared. Next, set the refinement parameters. Pull down the **Mesh** menu and select the **Refinement Impurities...** item. This panel controls which areas have a finer mesh than the others, based on the value of the impurity.

Note: Refining on impurities only takes place in the Semiconductor region.

9.10.3: Refining On Impurities

To cause refining for all doping, especially at impurity junctions, pull down the **Add** menu and select **Net Doping**. Next, to cause a reasonable number of triangles in the channel under the gate, again pull down the **Add** menu and this time choose boron. To make more triangles, the sensitivity of boron should be set to 0.2 on the lower part of the **Refinement Impurities** panel. The lower the number, the more triangles that are generated.

9.10.4: Mesh Constraints

The final operation is to set the mesh constraints. To do this, pull down the **Mesh** menu and select **Mesh Constraints**. In this example, it is assumed that obtuse triangles are acceptable in all regions except in the semiconductor. First, select the **All Regions** and set the **Max. Angle** to 180, either using the slider or entering in the value. This causes generation of far fewer triangles and points. However, because most simulators give poor results, or no results at all, if there are obtuse triangles in the semiconductor, the semiconductor regions need to have a stronger restriction. Select **Semiconductor Regions** and observe that the **Max. Angle** reads 180.0. To override the **All Regions** setting, select the check box to the left of **Max. Angle** and set it to 90.0. (This value can be overridden in a specific semiconductor region by selecting that region explicitly and setting its values).

9.10.5: Final Meshing

At this point DEVEDIT is ready to mesh, click on **Done**. To mesh, pull down the **Mesh** menu and select **Meshbuild**. Since **MeshBuild** is the default action on the **Mesh** menu, it can also be selected by simply clicking on **Mesh**. DEVEDIT now produces a mesh. A **Cancel** button is shown during meshing in case an unreasonable meshing parameter was supplied and it is readily apparent that too many triangles are being produced.

9.10.6: Saving the Silvaco Standard Structure File

To save this file, pull down the **File** menu and choose **Save as....** Specify a name to save the file (such as **test.str**) and click on the **SILVACO Standard Structure File** button. Then click on **Save File** to produce a file readable by all Silvaco 2-D Simulators.

9.11: IMPURITIES

9.11.1: Viewing Impurities

Currently, only net doping can be viewed. To view net doping, pull down the **Show Net Doping** menu on the base window. This menu controls pixel resolution of the displayed net doping contours. To display the net doping legend, pull down the **Net Doping Legend** menu and select a location to place the legend. (Doping at $1e+10$ level is considered to be approximately zero for graphing purposes).

9.11.2: Impurity Definition

Impurities can be read in when a structure file is loaded, or defined using a source line or box and rolloff function or profile. A structure file is a 2-D or 3-D impurity profile that is used to extrapolate on to the final mesh (these do not show up in the **User Impurities** panel list).

A new impurity distribution can be easily added by performing the following steps:

1. Enter the add impurity mode.
2. Define a impurity source area.
3. Define the rolloff directions.
4. Define the rolloff functions.
5. Define a join function.

9.11.3: Impurities Loaded From A Structure

When a structure file is loaded, the complete mesh is temporarily saved. When the device is saved as a structure file, each new mesh point's value is found by extrapolating the value based on the mesh point's material and the values from the closest triangle or prism in the original mesh. These values are added to any user created impurities to create the final impurity profile.

9.11.4: Add Impurity Mode

To add an impurity distribution, pull down the **Impurities** menu on top of the screen and Select **Add Impurity...** (that is the default action for **Impurities**). This action places the **Add Impurity** panel below.

Add Impurity

Impurity

Color

Draw Mode

Start X: Microns

Start Y: Microns

End X: Microns

End Y: Microns

Peak Concentration:

Reference Value:

Join Function

Distance:

Distance:

Figure 9-32: Add Impurity Panel

Note: Add can not be selected if an impurity is currently being modified until the modification is completed. If an impurity is being added, the Add Impurity panel will be displayed in its current state.

9.11.5: Defining An Impurity Source Area

The impurity source area is the area in which the impurity is set to a **Peak Concentration**. The area can be a box, a rectangle, a vertical line, a horizontal line, or point. For this area, there are 4 (or 6 in 3-D mode) possible rolloff directions as follows:

- above the area (-Y)
- below (+Y)
- left (-X)
- right (+X)
- front (-Z) for 3D
- back (+Z) for 3D

These are described more fully in the Defining Impurity Rolloff Direction section below.

To define the impurity source area, draw a rectangle on the existing device. To place the first corner, press and release the left mouse button over the desired location. Now move the mouse to the opposite corner, press and release the left mouse button again. To change the shape, move the mouse to the corner you wish to move, then press and hold the left mouse button. Move the pointer to the new position and release the button. Observe that the corner is placed. (Changing **Draw Mode** to **Line** forces the drawing to be either a horizontal or vertical line).

You can alternatively set the locations manually by entering values into the fields **Start X**, **End X**, **Start Y**, and **End Y**. In 3-D mode, the fields **Start Z**, and **End Z** must be entered in manually. This feature is also useful for more exact placement.

9.11.6: Defining Impurity Roll-off Direction

After defining an impurity source area, you must define how the impurity decreases as the distance from the source area increases along each axis. The first step in this procedure is to select the appropriate icon desired. If the impurity source area is a rectangle or a vertical line, the **Y Rolloff** icon choices are as seen in Figure 9-33. This is the case if the **Start Y** value is less than the **End Y** value. In this case, the detailed description for each icon can be found by locating the icon in column 1 of Figures 9-33 through 9-33a.

If the impurity source area is a horizontal line or a point, the **Y Rolloff** icon choices are as seen in Figure 9-34. This is the case if the Start Y value equals the End Y value. In this case, the detailed description for each icon can be found by locating the icon in column 2 of Figures 9-33 through 9-33.

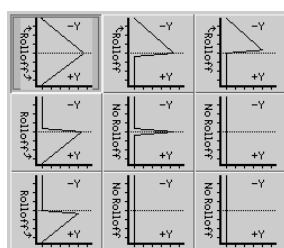


Figure 9-33: Rolloff Icon
Start Y \neq End Y

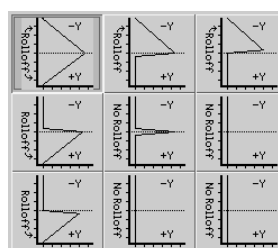


Figure 9-33a: Rolloff Icon
Start Y = End Y

If the impurity source area is a rectangle or a horizontal line, **X Rolloff** icon choices are as seen in Figure 9-34. This is the case if the **Start X** value is less than the **End X** value. In this case, the detailed description for each icon can be found by locating the icon in column 3 of Figures 9-34 through 9-34a.

If the impurity source area is vertical line or a point, the **X Rolloff** icon choices are shown in Figure 9-34a. This is the case if the **Start X** value equals the **End X** value.

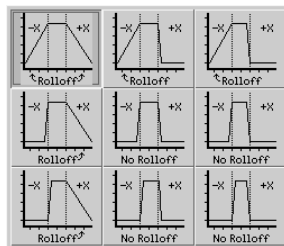


Figure 9-34: Rolloff Icon
Start X \neq End X

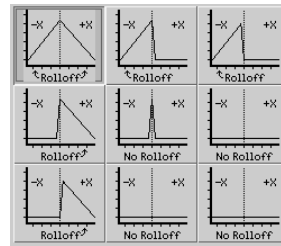


Figure 9-34a: Rolloff Icon
Start X = End X

Alternatively, the source area can radiate out of a circle. This is mainly used in 3D mode.

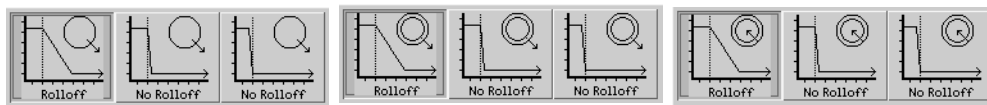


Figure 9-35: Circular

Rolloff=Both

Rolloff=Both (Figure 9-36) causes the same rolloff function to be used in both the negative and positive directions away from the impurity source area. While each axis has its own roll off function, to create different rolloffs in the negative and positive directions on the same axis, see Combining impurity rolloffs

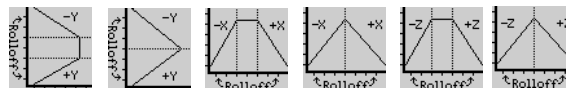


Figure 9-36: Rolloff=Both

Rolloff=High

Rolloff=High (Figure 9-37) causes the rolloff function to be used in the positive direction from the impurity source area. In the negative direction, the impurity value drops to (steps down to) zero if the distance is greater than zero.

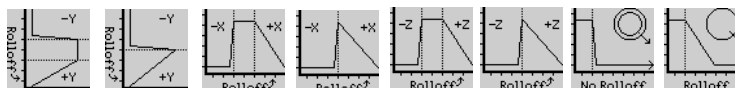


Figure 9-37: Rolloff=High

Rolloff=High.P.Step

Rolloff=High.Premature.Step (or **High.P.Step**) (Figure 9-38) causes the rolloff function to be used in the positive direction from the impurity source area. In the negative direction, the impurity value drops to (steps down to) zero if the distance is greater or equal to zero. This causes the negative edge of the impurity source area to have a zero value. As you can see, if the start value along the axis equals the end value, the entire impurity source area will have a zero value.

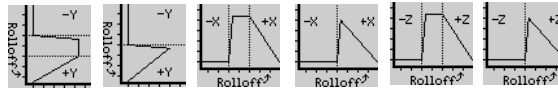


Figure 9-38:Rolloff=High.P.Step

Rolloff=Low

Rolloff=Low (Figure 9-39) causes the rolloff function to be used in the negative direction from the impurity source area. In the positive direction, the impurity value drops to (steps down to) zero if the distance is greater than zero.

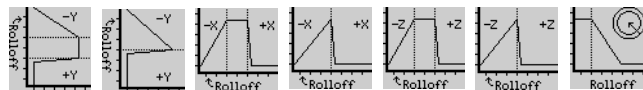


Figure 9-39:Rolloff=Low

Rolloff=Step

Rolloff=Step (Figure 9-40) causes no rolloff function to be used. The impurity source area contains the peak value, which immediately steps down to zero away from the impurity source area in the given direction.

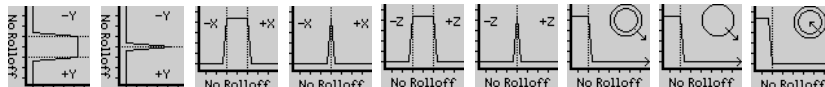


Figure 9-40:Rolloff=Step

Rolloff=Step.P.Low

Rolloff=Step.Premature.Low (or **Step.P.Low**) (Figure 9-41) causes no rolloff function to be used. The impurity source area contains the peak value except along the negative edge, which is zero. All areas outside the impurity source area also get zero values. As you can see in columns 2 and 4 of Figure 9-43, this is completely useless if Start X/Y equals End X/Y.

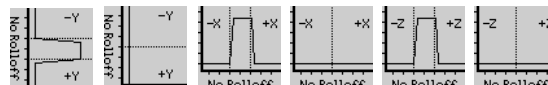


Figure 9-41:Rolloff=Step.P.Low

Rolloff=Low.P.Step

Rolloff=Low.Premature.Step (or **Low.P.Step**) (Figure 9-42) causes the rolloff function to be used in the negative direction from the impurity source area. In the positive direction, the impurity value drops to (steps down to) zero if the distance is greater or equal to zero. This causes the positive edge of the impurity source area to have a zero value.

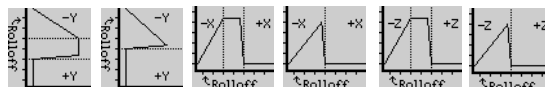


Figure 9-42:Rolloff=Low.P.Step

Rolloff=Step.P.High

Rolloff=Step.Premature.High (or **Step.P.High**) causes no rolloff function to be used. The impurity source area contains the peak value except along the positive edge, which is zero. All areas outside the impurity source area also get zero values. As you can see in columns 2 and 4 of Figure 9-43, this is completely useless if Start X/Y equals End X/Y.

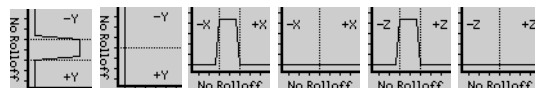


Figure 9-43:Rolloff=Step.P.High

Rolloff=P.Step

Rolloff=Premature.Step (or **P.Step**) causes no rolloff function to be used. The impurity source area contains the peak value except along the edges, which are zero. All areas outside the impurity source area also get zero values. As you can see in columns 2 and 4 of Figure 9-44, this is completely useless if Start X/Y equals End X/Y.

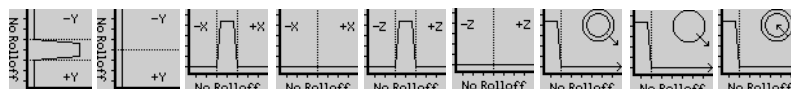


Figure 9-44:Rolloff=P.Step

9.12: ROLL-OFF FUNCTION

The **Roll-off** function calculates the vertical and the horizontal rolloffs separately and then uses a join function if both a vertical distance and a horizontal distance exists. There are two types of rolloff functions, analytic and profiled. When selecting a rolloff function, the first column of choices are the analytic functions described below. The next column lists you added doping profiles. These are described just after the analytic functions.

9.12.1: Analytic Functions

Analytic functions are used to describe the rolloff curve from the impurity source area. The function is given a distance (distance). This distance is computed by the join function. The join function splits the mesh points distance from the source area into two components: the (X rolloff distance) and the (Y rolloff distance). How (distance) is computed and how (doping%) is used is dependent on the function and is described in that section.

Note: Functions ending in **(Dist)** will have Reference Value concentration when (distance) equals the user-supplied **Distance**.

User Supplied Variables

Peak Concentration: P
 Constant: K
 Reference Value: R
 Distance: D

Location Dependent Variables

Distance from Base (X,Y, or Z) d
 X Distance from Base dx
 Y Distance from Base dy
 Doping % for the current direction p

- Gaussian

$$dK = \frac{d}{K} \quad 9-1$$

$$p = e^{\left(\frac{-dK}{2}\right)} \quad 9-2$$

- Gaussian (Dist)

$$\text{First solves for } dk: \frac{R}{P} = e^{\frac{-d_k^2}{2.0}} \quad 9-3$$

$$p = e^{\left(\frac{-d_k^2}{2}\right)} \quad 9-4$$

- Error Function

$$d_K = \frac{d}{K} \quad 9-5$$

$$p = e^{-\left(1.02d_K + 0.79276d_K^2 + 0.019345d_K^3\right)} \quad 9-6$$

- Error Function (Dist)

First solves for dk : $\frac{R}{\bar{P}} = e^{-\left(1.02d_k + 0.79276d_k^2 + 0.019345d_k^3\right)}$ 9-7

$$p = e^{-\left(1.02d_k + 0.79276d_k^2 + 0.019345d_k^3\right)} \quad 9-8$$

- Linear (Dist)

$$k = \frac{1.0 - \frac{R}{\bar{P}}}{D} \quad 9-9$$

$$d_k = d \bullet k \quad 9-10$$

$$p = 1.0 - d_k \quad 9-11$$

Note: If $p < 0$ then $p = 0$.

- Logarithmic

$$d_K = d \bullet K \quad 9-12$$

$$p = \frac{1.0}{\log(d_K) + 1.0} \quad 9-13$$

- Logarithmic (Dist)

$$k = e^{\left(\frac{\frac{P}{\bar{R}} - 1.0}{D}\right)} \quad 9-14$$

$$d_k = d \bullet k \quad 9-15$$

$$p = \frac{1.0}{\log(d_k) + 1.0} \quad 9-16$$

- Exponential

$$d_K = d \bullet K \quad 9-17$$

$$p = \frac{1.0}{e^{(d_k + 1.0)}} \quad 9-18$$

- Exponential (Dist)

$$k = \frac{\log\left(\frac{P}{R} - 1.0\right)}{D} \quad 9-19$$

$$d_k = d \bullet k \quad 9-20$$

$$p = \frac{1.0}{e^{(d_k + 1.0)}} \quad 9-21$$

- Constant

$$p = 1.0(100percent) \quad 9-22$$

- Step Function

Deprecated Option. Use constant rolloff and rolloff icons instead of the step function. This option will be removed in future releases.

9.12.2: Doping Profiles

Doping profiles are a list of distances and impurity concentrations at those distances. These can be obtained from SSUPREM3 structure files, SSUPREM4 1-D structure files, and the SPDB doping database. By using these values and one of the four extrapolation functions, a user can add an impurity using this profile as a rolloff function instead of one of the analytical profiles contained in DEVEDIT.

Adding a New Doping Profile

To add a new doping profile, using the **Impurities** menu, choose **Doping Profiles....** The **User Defined Doping Profiles** control panel is displayed on the right side of the screen. The top list shows the doping profiles already read into DEVEDIT. Below is information about the selected doping profile. The name and impurity type can be changed in this area. To read in a SSUPREM3 structure file or a 1-D ATHENA structure file, click on the **Load File...** button. A popup window appears. Select a file and load it. A popup notice is then be displayed asking which impurity you wishes to load from that file. Now the profile is loaded and given the name NewProfile001 or NewProfile 002, or the first new not already in use.

Note: If the impurity selected is “active antimony”, “active arsenic”, “active boron”, or “active phosphorus”, DevEdit will store these as “antimony”, “arsenic”, “boron”, or “phosphorus”, respectively.

Back in the **User Defined Doping Profiles** command panel, this profile is added to the **User defined profiles** list and becomes the selected item. Details about this profile are displayed below the list. You can now rename the profile. Replace the name in the **Profile Name:** field with the desired name. Be careful not to accidentally use a name that already exists in the list. You can now click on **Done** to remove the **User Defined Doping Profiles** command panel. This doping profile has now been added to the possible rolloff functions in the **Add Impurity** panel and the **Modify Impurity** panel.

If a user defined profile is selected as the rolloff function in the **Add Impurity** panel or the **Modify Impurity** panel. The peak value of that impurity distribution is set to the value of the doping profile at distance equal zero. The impurity is also set to match the profile. Both fields are grayed out as long as a doping profile is being used as a rolloff function. While different doping profiles can be used in the X and Y directions, they must be compatible; i.e., have the same value at distance equals zero and be the same impurity type.

9.12.3: Join Function

There are currently three join functions. First, the multiple join works by computing the Y rolloff and applying the X rolloff. This effectively is a multiply of the two rolloffs. Second, the interpolate join works by considering an arc at equal distances between the Y rolloff and the X rolloff though the locations and interpolating the values along the arc. Third, the miter join takes the lower value after allowing for the X rolloff or the Y rolloff.

User Supplied Variables

- Multiply Join

Peak Concentration: P

d_y = Y rolloff distance = depth distance = Δy

d_x = X rolloff distance = lateral distance = Δx

$$doping = p_x \bullet p_y \bullet P \quad 9-23$$

- Interpolate Join:

d = Y rolloff distance = X rolloff Distance = total= depth distance

$$d = \sqrt{\Delta x^2 + \Delta y^2} \quad 9-24$$

$$d_y = d \quad 9-25$$

$$d_x = d \quad 9-26$$

$$doping = \left[\left(p_x \bullet \frac{\Delta x^2}{d} \right) + \left(p_y \bullet \frac{\Delta y^2}{d} \right) \right] \bullet P \quad 9-27$$

- Miter Join:

- d_y = Y rolloff distance = depth distance = Δy

- d_x = X rolloff distance = lateral distance = Δx

- if $p_y < p_x$ then

$$doping = p_y \bullet P \quad 9-28$$

else
 $doping = p_x \bullet P$

9-29

To add these changes to the device, click on **Apply**. Click on **Cancel** if you do not want to add or modify this impurity.

9.12.4: Deleting Impurities

To delete an analytic impurity, select an impurity in the **User Defined Impurity** list on the main panel; pull down the **Impurities** menu and choose **Delete**.

9.12.5: Editing Impurities

To edit an analytic impurity, select an impurity in the **User Defined Impurity** list on the main panel; pull down the **Impurities** menu, and select **Modify**. The same panel used for adding an impurity is now displayed with all the filled in values. These values can now be edited.

9.12.6: Combining Impurity Rolloffs

In the special case where the rolloff in one direction along an axis is different from the rolloff in the other direction along the same axis, two impurity specifications must be combined to produce one impurity profile. Special care must be taken when defining these impurity definitions to prevent similarities in the seam between the two impurity definitions. This could cause the impurity to be zero or twice the “Peak concentration” at the seam.

The following is an example of combining two impurity definitions to simulate the surface effect of a doping implant:

1. Enter DEVEDIT and set the work area to 0,-0.25 and 2,2.
2. Create two regions. The first region is silicon with corners at 0,0 2,0, 2,2, and 0.2, and a base of boron of $1e+16 /cm^3$. (Refer to Section 9.6.1: “Adding a Region” for more details). The second region is aluminum with corners at 0,0, 1,0, 1,-0.2.
3. Create an impurity distribution to describe the rolloff in the down direction, the lower half the desired profile.
4. Click on the **Impurities** button at the top of the screen (see Figure 9-45) and then set the following parameters:
 - Impurity: Arsenic
 - Start X: 0
 - Start Y: 0.2
 - End X: 1
 - End Y: 0.2
 - Peak Concentration: $1e+20$
 - Reference Value: $1e+16$ (base impurity value of silicon)
 - Rolloff = High Y Rolloff: Gaussian (Dist)
 - Distance: 1 (distance to junction)
 - Rolloff = Both X Rolloff: Error Function
 - Constant: 0.1
5. Press **Apply**.

Add Impurity

Impurity

Color

Draw Mode

Start X: Microns

Start Y: Microns

End X: Microns

End Y: Microns

Peak Concentration:

Reference Value:

Join Function

☒ Y Rolloff: Distance:

☒ X Rolloff: Distance:

Figure 9-45: Downward Impurity Rolloff

6. To add the other half, again click on the **Impurities** button (see Figure 9-46) and set the following parameters:

- Impurity: Arsenic
- Start X: 0
- Start Y: 0.2
- End X: 1
- End Y: 0.2
- Peak Concentration: $1e+20$
- Reference Value: $1e+16$ (impurity value at surface of silicon)
- Rolloff = High Y Rolloff: Gaussian (dist)
- Distance: 1 (distance to junction)
- Rolloff = Both X Rolloff: Error Function
- Constant: 0.1

7. Press Apply.

Add Impurity

Impurity

Color

Draw Mode

Start X: Microns

Start Y: Microns

End X: Microns

End Y: Microns

Peak Concentration:

Reference Value:

Join Function

☒ Y Rolloff: Distance:

☒ X Rolloff: Distance:

Figure 9-46: Upward Impurity Rolloff

Now you can create a mesh and save the file as a **Silvaco Standard Structure File** named `example2.str` (perform steps in the Subsections “Mesh Creation” on page 9-15 and “Saving The File” on page 9-17). Figure 9-47 shows the net doping contours on the created device, using TONYPLOT. A cut line is made in the vertical direction shown in Figure 9-48.

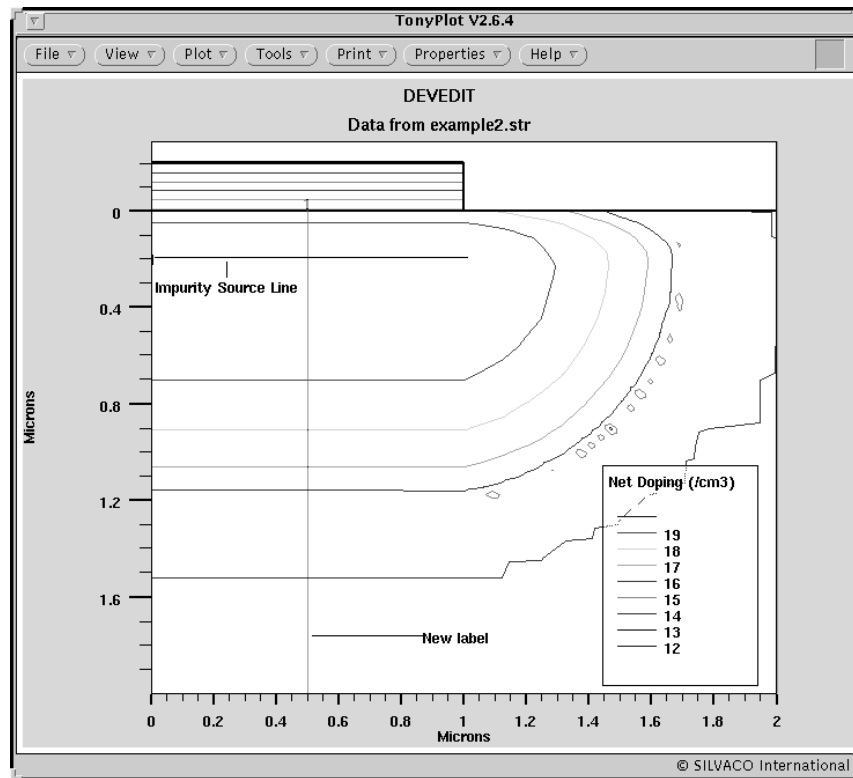


Figure 9-47: Net Doping Contour on Created Device using TonyPlot

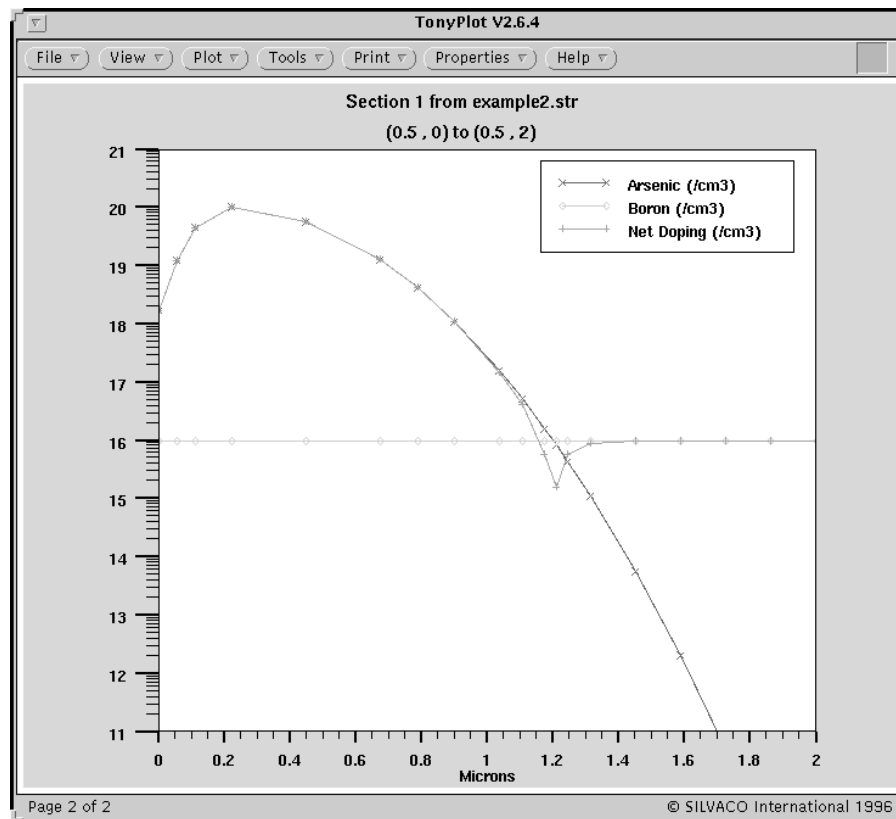


Figure 9-48: Cutline Made in the Vertical Direction

9.13: STATEMENTS

9.13.1: Overview

This chapter contains a complete description (in alphabetical order) of every statement and parameter used by any of the DEVEDIT products. The following documentation is provided for each statement:

- The statement name
- The syntax of the statement
- A list of all of the parameters of the statement and their type
- A description of each parameter or group of similar parameters
- An example of the correct usage of each statement

9.13.2: Cards And Parameters

Card Syntax

PARAM = required parameter

[PARAM] = optional parameter

<N> = integer or floating point number

<C> = user defined string can be quoted by 'or "

<BOOLEAN> = the strings true, false, yes, no, on or off. In the case param, param can be used for param=true and !param or ^param can be used for param=false.

<POINT2D> = a 2d point (ex. 0.5,3.0) note: Points are in Microns

<POINT2D_LIST> = a list of 2d points (ex. "0,0 1,0 1,1 0,1 0,0") quotes must be used if more than one point is in list

{PARAM1 PARAM2} = parameters are associated with each other

PARAM1 | PARAM2 = must supply param1 or param2 but not both

PARAM1 || PARAM2 = must supply param1 or param2 or both

PARAM... = param repeated one or more times

[PARAM]... = param repeated zero or more times

Examples

In the following example param1 or param3 must be supplied. If, and only if, param1 is supplied an optional param2 can be supplied. In any case, at least one param4 must be supplied:

```
card {param1=<n> [param2=<c>]} | param3 param4=<n>...
```

Line Continuation

Cards can be continued across multiple lines by ending the line with a backslash (\). In this case, the next line is considered part of the same line.

Note: When loading a command file, lines can also be continued by starting the next line with a plus sign (+). This function has been depreciated and is only supplied for backwards compatibility and can be removed in future versions.

Comments

Comments can be placed at the end of any card or on lines by them selves. Comments start with a number sign (#) and end at the end of the line, regardless whether or not the line ends with a backslash (\). Depreciated function: lines starting with \$ are also considered comments. This is for backwards compatibility and can be removed in future releases.

Parameter Section

In the parameter section for each card, each paragraph starts with a parameter's full name and value type. Following this there can be a list of alternate parameter names. This list is contained in parenthesis. Any length abbreviation for a card or parameter can be used as long as it uniquely identifies the card or parameter. However, a reasonable length should be used to keep names from becoming ambiguous when a newer version is installed.

9.13.3: BASE.MESH

Starting point when generating a new mesh.

Syntax

```
BASE.MESH [HEIGHT=<N>] [WIDTH=<N>]
```

Description

When a mesh is created with the mesh card and the mode is mesh.build, mesh build creates a base tensor product mesh and refines the geometry, impurities, and mesh constraints. The base mesh may not be regular to allow for the geometry of the device. The original default for width and height is 100000 microns, which means the structure will have 4, 6 or 9 squares in the base mesh, no matter the device size.

Parameters

HEIGHT=<n> (*h,y*) Maximum height (Δy) of each rectangle in the base mesh in microns.

WIDTH=<n> (*w,x*) Maximum width (Δx) of each rectangle in the base mesh in microns.

Replaces Card

```
BaseMesh [Height=<n>] [Width=<n>]
```

See Also

MESH, CONSTRAIN.MESH

9.13.4: BOUNDARY.CONDITIONING

Reduce number of boundary points.

Preferred Abbreviation

bound.cond or bnd.cond

Syntax

```
BOUNDARY.CONDITIONING[ [WHEN=] <C> ] [ [WHEN=] NEVER | ONCE | AUTOMATIC ] \
[ MAXIMUM.SLOPE=<N> ] [ MAXIMUM.RATIO=<N> ] [ ROUNDING.UNIT=<N> ] \
[ LINE.STRAIGHTENING=<N> ] [ ALIGN.POINTS[=<BOOLEAN>] ]
```

Description

Set or apply boundary conditions to limit complexity of borders between region. This may help limit the number of triangles creating during mesh creation. This may destroy the existing mesh.

Parameters

[WHEN=] never | once | automatic (default = automatic) When boundary conditions will be performed.

Never = Turns off boundary conditioning.

Once = Performs boundary conditioning now (when card is read).

Automatic = Performs current conditioning and before each mesh command.

MAXIMUM.SLOPE=<n> (max.slope) The maximum ratio of the vertical height to the horizontal width of each boundary segments. If the ratio is greater than this, it is broken into two line segments; one vertical or horizontal and one with this limit.

Note: Vertical and horizontal lines are considered to have a zero ratio, not an infinite one. Therefore, only lines close to being horizontal and vertical lines are affected. This is used to limit the number of triangles mesh.build creates. This number must always be less than the maximum triangle ratio (max.ratio).

MAXIMUM.RATIO=<n>] (MAX.RATIO) The maximum ratio of a triangles height to its width.

ROUNDING.UNIT=<n> (RND.UNIT, RND) All boundary points are rounded to an even multiple of this unit.

Note: Points created by mesh.build are not rounded to this unit but will be strongly affected by the initial boundary point locations.

LINE.STRAIGHTENING=<n> (LINE.STR) If two boundary segments have a joining angle of greater than or equal to 180-line.straightening, the two line segments are combined by removing the joining point.

ALIGN.POINTS[=<BOOLEAN>] If a boundary point joins two almost horizontal lines, the point can be moved slightly in the horizontal direction to align it with other points. This is also true in the vertical direction.

Replaces Card

```
BoundaryConditioning [AutoConditioning=<c>] [MaxSlope=<n>] \
[MaxRatio=<n>] [RoundingUnit=<n>] [LineStraightening=<n>] \
[AlignPoints[=<boolean>]] [NoSet] [NoApply]
```

See Also

MESH, CONSTRAIN.MESH

9.13.5: CONSTRAINT.MESH

Set limits (constraints) on triangles created during mesh and refine operations.

Preferred Abbreviation

`constr.mesh`

Syntax

```
CONSTRAIN.MESH [GLOBAL] [REGION.ID=<N>] [REGION.NAME=<C>] \  
[MATERIAL=<C>] [MATERIAL.TYPE=<C>] [x1=<N> y1=<N> x2=<N> y2=<N>] \  
[UNDER.REGION=<C> | UNDER.MATERIAL=<C> | UNDER.GATE DEPTH=<N>] \  
[DEFAULT] [MAXIMUM.ANGLE[=<N>]] \  
[MAXIMUM.RATIO[=<N>]] [MAXIMUM.ADJACENT[=<N>]] \  
[MAXIMUM.HEIGHT[=<N>]] [MAXIMUM.WIDTH[=<N>]] \  
[MINIMUM.HEIGHT[=<N>]] [MINIMUM.WIDTH[=<N>]]
```

Description

Mesh constraints are used to determine the size and shapes of triangles during the meshing phase. Weaker constraints create a less dense mesh and may be used to improve execution time in subsequent process/device simulations. The constraints are arranged in a hierarchy; global constraints, material type constraints, and region specific constraints. To determine the active constraint in a region, DEVEDIT starts with the global value. This value can be overridden by a material type constraint which in turn can be overridden by a region specific constraint.

When DEVEDIT is started all material type constraints default to the global constraints. The global constraints have predetermined default values. As new regions are added, the regions initially default to the material type constraint associated with the regions material. Once a value has been set, it can be restored to default to the more general level by using a parameter with no value. For example, `MAX.ANGLE=100` can later be cleared by using `MAX.ANGLE`. All values for the current constraint(s) can be cleared by using the `DEFAULT` parameter (see Figure 9-49).

In addition to this constraint hierarchy, there may be rectangular based constraints. This areas can be specified with use absolute coordinates, using `x1`, `y1`, `x2`, and `y2`. The rectangular area can also be implied in a semiconductor area by using `under.region`, `under.material`, or `under.gate`, AND setting depth.

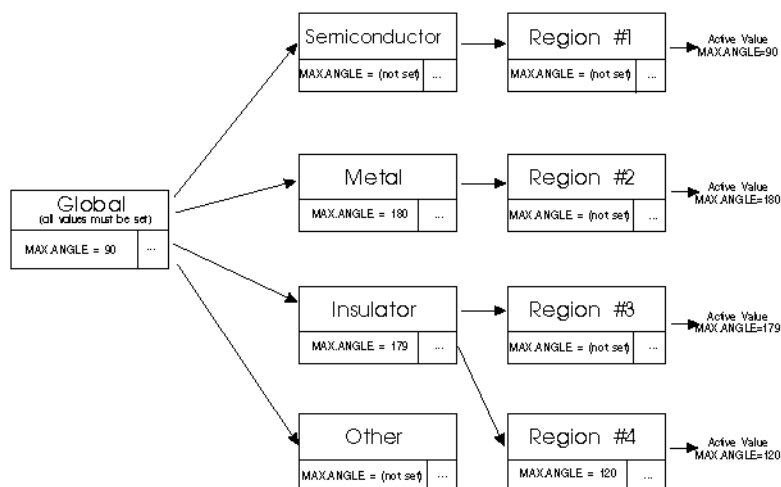


Figure 9-49:Constraint Hierarchy

Parameters

GLOBAL sets global constraints. The global constraints is also if region and material type are NOT used.

REGION.ID=<n> (REG) specifies which region (by region id number) these constraints apply to. Several region parameters can be set at the same time.

REGION.NAME=<c> (REG) specifies which region (by name) these constraints apply to. Several region parameters can be set at the same time.

MATERIAL.TYPE=<c> (MAT.TYPE, TYPE) Specifies what material types these constraints apply to. Values can be: Semiconductor, Metal, Insulator, Other.

x1=<N> y1=<N> x2=<N> y2=<N> The four corners of a rectangular area to which the following constraints will apply.

UNDER.REGION=<C>> In the semiconductor region under this region, a rectangle is defined from the surface to the depth specified using the depth parameter. If the region is disjoint, multiple rectangles can be formed.

UNDER.MATERIAL=<C> In the semiconductor region under regions of this material, a rectangle is defined from the surface to the depth specified using the depth parameter. If multiple regions are made of the same material, multiple rectangles can be formed. Values can be Poly, Aluminum, SiO2, etc. See Generic Parameters - Material for a more complete list.

UNDER.GATE This is currently the same as “under.mat=poly”. It may become more intelligent about finding the gate in future releases.

DEPTH=<N>> How deep (in Microns) the rectangle should be from the surface of the semiconductor region. This parameter must be used in conjunction with under.region, under.material, or under.gate.

DEFAULT Before setting specified values, reset all values in the specified region, material type, or global constraints to their default values.

MAXIMUM.ANGLE[=<n>] (MAX.ANGLE) Maximum angle a triangle can contain. Max.angle has a range between 90 and 180 degrees. Normally points are limited by mesh.build to 12 connections. If the maximum angle is set to 180, the connections limit is removed. In no case will an angle actually be 180 degrees.

MAXIMUM.RATIO[=<n>]] (MAX.RATIO) Maximum ratio of a triangles height to its width.

MAXIMUM.HEIGHT[=<n>] (MAX.H) Maximum height of a triangle in microns.

MAXIMUM.WIDTH[=<n>] (MAX.W) Maximum width of a triangle in microns.

MINIMUM.HEIGHT[=<n>] (MIN.H) Triangles shorter than this are not shortened during impurity refinement.

MINIMUM.WIDTH[=<n>] (MIN.W) Triangle narrower than this are not narrowed during impurity refinement.

EXAMPLES

```
# allow all regions to have slightly obtuse triangle
# (sets global constraints) constr.mesh max.angle=100
# allow only non-obtuse triangles in semiconductor regions
constr.mesh mat.type=semiconductor max.angle=90

# allow region #1 to have slightly obtuse triangle even
# if region is a semiconductor.
constr.mesh reg=1 max.angle=95

# now let region #1 default to it material.type or the
# global constraints
const.mesh reg=1 max.angle

# clear all current setting in region #3 and the metal material
# type constraints and set possible triangles to very obtuse
const.mesh reg=3 mat.type=metal default max.angle=180

# Make sure the triangle in the channel of a mos device
# has small enough triangle (assuming the channel is directly
# below a region named "gate").
const.mesh under.region=gate depth=0.5 max.height=0.1 max.width=0.25

# Make sure all contacts have enough connecting point for simulation.
const.mesh under.mat=aluminum depth=0.0001 max.width=0.25
```

Replaces Card

```
ConstrainMesh [Region=<n> | Material=<c> | Type=<c>] \
  [[!^]MaxAngle[=<n>] [[!^]MaxRatio[=<n>]] \
  [[!^]MaxAdjacent[=<n>]] [[!^]MaxHeight[=<n>]] \
  [[!^]MaxWidth[=<n>]] [[!^]MinHeight[=<n>]] \
  [[!^]MinWidth[=<n>]]
```

See Also

IMPURITY.REFINE, MESH

9.13.6: CUT

Cut out a strip from the device and join the two pieces together.

Syntax

```
CUT {{X1=<N> X2=<N>} || {Y1=<N> Y2=<N>}} [AUTOMATIC.JOIN[=<BOOLEAN>]]
```

Description

Cut out a vertical strip between $x=x1$ and $x=x2$, or cut out a horizontal strip between $y=y1$ and $y=y2$, or both. Points at $x1/y1$ will not be moved. Points at $x2/y2$ will be moved to $x1/y1$ dragging point beyond $x2/y2$ with them.

Parameters

X1=<n> Start of x direction cut.

X2=<n> End of x direction cut.

Y1=<n> Start of y direction cut.

Y2=<n> End of y direction cut.

AUTOMATIC.JOIN[=<boolean>] (AUTO.JOIN) If two regions made of the same material presently touch, the regions are joined into one region. The attributes from the region with the lowest id will be used for the combined region. The default value is true.

9.13.7: DEPOSIT

Deposit a layer of material.

Syntax

```
DEPOSIT MATERIAL=<C> THICKNESS=<N> [ROUNDING.ANGLE=<N>] \
  [[SIDE=]TOP|LEFT|RIGHT|BOTTOM] [START=<N>] [END=<N>] \
  [REGION.ID=<N>] [REGION.NAME [COLOR=<N>] [PATTERN=<N>] \
  AUTOMATIC.JOIN[=<BOOLEAN>]]
```

Description

Deposit a uniform thickness of a material on the specified side.

Parameters

[SIDE=]TOP|LEFT|RIGHT|BOTTOM Side on which to deposit the new region. The default is side=top.

THICKNESS=<n> Thickness (in microns) of the deposit.

START=<n> The start of the deposit. If side=top or side=bottom start is an *x* coordinate, otherwise it is a *y* coordinate. Default value is left side of structure or top of structure, respectively.

END=<n> The end of the deposit. If side=top or side=bottom, start is an *x* coordinate, otherwise it is a *y* coordinate. Default value is right side or bottom of structure, respectively.

ROUNDING.ANGLE (RND.ANGLE) When making a corner during a deposit, this angle is used to determine how many points are used. The angle should be between 5 and 45 degrees and be an even divisor of 90. The default angle is 30 degrees.

REGION.ID=<n> (REG.ID) Region id number of the deposited region, if the region reg is not joined with an existing region. See auto.join.

REGION.NAME=<c> (REG.NAME) Region name of the deposited region, if the region reg is not joined with an existing region. See auto.join.

MATERIAL=<c> Specifies which material will be deposited (i.e., Silicon, Aluminum, AlGaAs, etc...). See Generic Parameters for a more complete description.

AUTOMATIC.JOIN[=<boolean>] (AUTO.JOIN) If two regions made up of the same auto.joinmaterial presently touch, the regions are joined into one region. Attributes from region with the lowest id will be used for the combined region. The default value is true.

COLOR=<n> (COLOUR) Color used to display region during DEVEDIT in X windows mode. This is a RGB bitmap with eight bits color. DEVEDIT has only a limited subset of these colors, therefore the closest match is used. Some basic colors can be specified by name; such as, red, green, blue, yellow, cyan, magenta, black, and white. See Generic Parameters for a more complete description.

PATTERN=<n> Fill pattern used to display region in DEVEDIT X windows mode. See Generic Parameters for a more complete description.

9.13.8: FLIP

Flip (make a mirror image of) the device.

Syntax

```
FLIP X=<N> | Y=<N>
```

Description

Flip (make a mirror image of) the device around a vertical or horizontal line or both.

Parameters

X=<n> Flip device around the vertical line *x*=<n>.

Y=<n> Flip device around the horizontal line *y*=<n>.

Examples

```
flip x=1
# point 0,0 becomes 2,0
# point 2,2 becomes 0,2
```



```
# point 4,4 becomes -2,4
flip y=0
```

```
# point 0,0 becomes 0,0
# point 2,2 becomes 2,-2
# point 4,4 becomes 4,-4
flip x=3 y=2
```

```
# point 0,0 becomes 6,4
# point 2,2 becomes 4,2
# point 4,4 becomes 2,0
```

9.13.9: IMPURITY

Add, replace or delete an impurity profile (analytic implant).

Syntax

```
IMPURITY {ID=<N> DELETE [REGION.ID=<N>]} \
| { [ID=<N>][REGION.ID=<N>] [IMPURITY=<C>] \
  { [PEAK.VALUE=<N>] | [RESISTIVITY=<N>] } \
  [REFERENCE.VALUE=<N>] [COLOR=<N>] [COMBINATION.FUNCTION=<C>] \
  [ { { [Y1=<N> Y2=<N> ROLLOFF.Y=<C> \
    [ CONCENTRATION.FUNCTION.Y=<C> \
      [ COEFFICIENT.Y=<N> | CONCENTRATION.PARAM.Y=<N> ] \
      [CONCENTRATION.SCALE.FACTOR.Y=<N>] ] } \
    || { X1=<N> X2=<N> ROLLOFF.X=<C> \
      [ CONCENTRATION.FUNCTION.X=<C> \
        [ COEFFICIENT.X=<N> | CONCENTRATION.PARAM.X=<N> ] \
        [CONCENTRATION.SCALE.FACTOR.X=<N>] ] } } \
  | { X=<N> Y=<N> R1=<N> ROLLOFF.R1=<C> \
    [ CONCENTRATION.FUNCTION.R1=<C> \
      [ COEFFICIENT.R1=<N> | CONCENTRATION.PARAM.R1=<N> ] \
      [CONCENTRATION.SCALE.FACTOR.R1=<N>] ] \
    [ R0=<N> ROLLOFF.R0=<C> \
      [ CONCENTRATION.FUNCTION.R0=<C> \
        [ COEFFICIENT.R0=<N> | CONCENTRATION.PARAM.R0=<N> ] \
        [CONCENTRATION.SCALE.FACTOR.R0=<N>] ] ] } \
  | { BASE1=<N>, <N> BASE2=<N>, <N> \
    ROLLOFF.Y=<C> \
      [ CONCENTRATION.FUNCTION.Y=<C> \
```

```
[ COEFFICIENT.Y=<N> | CONCENTRATION.PARAM.Y=<N> ] \
[CONCENTRATION.SCALE.FACTOR.Y=<N>] ] \
ROLLOFF.X=<C> \
[ CONCENTRATION.FUNCTION.X=<C> \
[ COEFFICIENT.X=<N> | CONCENTRATION.PARAM.X=<N> ] \
[CONCENTRATION.SCALE.FACTOR.X=<N>] ] } } ] \
[ { Z1=<N> Z2=<N> } | Z=<N>,<N> \
ROLLOFF.Z=<C> \
[ CONCENTRATION.FUNCTION.Z=<C> \
[ COEFFICIENT.Z=<N> | CONCENTRATION.PARAM.Z=<N> ] \
[CONCENTRATION.SCALE.FACTOR.Z=<N>] ] ] }
```

Description

Implant (add) an impurity or quantity to a device by using the impurity card. The model used assumes that a rectangle (or a box in 3-D mode) has a peak value and then rolls off from this peak using a vertical formula (y), a horizontal formula (x), and possibly a width formula (z). These formulae can be abruptly stopped in any of the six directions (up, down, left, right, forward, and backward) by using the rolloff mode. The rolloff mode specifies whether a roll-off function is used or where exactly the impurity is dropped to zero. See the Impurity chapter for a more detailed description.

Parameters (Z Parameters are only valid in 3D Mode)

ID=<n> is the identifier of which impurity “implant” should be added, replaced or deleted. If no id is given, the first unused id will be added.

Note: There is a list of id's for all regions and a separate list for each region. (i.e. there can be an impurity #1 for all regions, an impurity #1 for region #5 and an impurity #1 for region #7, which make up three different impurities.)

DELETE Deletes the impurity “implant” identified by id.

REGION.ID=<n> is the region number which uniquely identifies the region reg.id to which region this impurity applies. If no **region.id** parameter is given (the normal case), it applies to all regions.

REGION.ID=<c> (REG.ID) Identifies the region with name <c> to which this impurity applies. If two regions have name <c>, the region with the lowest region number will be replaced or deleted.

REGION.NAME=<c> (REG.NAME) Identifies the region with name <c> to which this impurity applies. If multiple regions have name <c>, all matched regions are changed.

IMPURITY=<c> Specifies what impurity this profile is describing (i.e., Boron, Arsenic, Potential, etc...) See Generic Parameters for a more complete description.

PEAK.VALUE=<n> Value of impurity in the base box. When using a 1D profile, the peak value in the profile will be linearly scaled to match this peak value.

REFERENCE.VALUE=<n> (REF.VALUE) Value of impurity at the given distance. See concentration.function and concentration.coefficient for more information.

X1=<n> Left side of base box. (In microns)

X2=<n> Right side of base box. (In microns)

Y1=<n> Top of base box. (In microns)

Y2=<n> Bottom of base box. (In microns)

Z1=<n> Front of base box. (In microns)

Z2=<n> Back of base box. (In microns)

BASE.1=<n>,<n> This is the “left, top” corner of the peak impurity rectangle. This parameter is depreciated. Use x1 and y1 instead.

BASE.2=<n>,<n> This is the “right, bottom” corner of the peak impurity rectangle. This parameter is depreciated. Use x2 and y2 instead.

COMBINATION.FUNCTION=<c> This describes how the x, y and z comb.funcrolloffs intersect. Possible values are: multiply, interpolate, or miter.

ROLLOFF.Y=<c>, ROLLOFF.X=<c>, and ROLLOFF.Z=<c> Possible values are:

- both
- high
- high.premature.step (high.p.step)
- low
- step
- step.premature.high (step.p.high)
- low.premature.step (low.p.step)
- step.premature.low (step.p.low)
- premature.step (p.step)

CONCENTRATION.FUNCTION.{Y|X|Z}=<c> (CONC.FUNC.{Y|X|Z}): Possible values are shown in the table below.

Warning: These values must match exactly. Do **NOT** abbreviate.

Full Name	Short Name
"Gaussian"	gauss
"Gaussian (Dist)"	gauss.dist
"Error Function"	erfc
"Error Function (Dist)"	erfc.dist
"Linear (Dist)"	dist
"Logarithmic"	log
"Logarithmic (Dist)"	log.dist
"Exponential"	exp
"Exponential (Dist)"	exp.dist
"Step Function"	obsolete use rolloff=step
<ld_profile_name>	

COEFFICIENT.{Y|X|Z}=<n>(CONC.COEF.{Y|X|Z}): (Concentration coefficient is valid only if concentration function is NOT a 1D profile). If the concentration function is a distance function, this is the distance (in microns) between the peak.value and the reference.value. Otherwise, it is a function specific coefficient.

CONCENTRATION.PARAMETER.{Y|X|Z}=<c>(CONC.PARAM.{Y|X|Z}): If the concentration function is a 1D profile, then it must be one of the following approximation functions. These functions specify the points between data points and at the end of the specified data.

Full Name	Short Name
"Log Extrapolate"	log.ex
"Log Interpolate"	log.in
"Linear Extrapolate"	lin.ex
"Linear Interpolate"	lin.in

If the concentration function is not a 1D profile, then `CONCENTRATION.PARAMETER.{Y|X|Z}` is an alias for `COEFFICIENT.{Y|X|Z}`.

CONCENTRATION.SCALE.FACTOR.{Y|X|Z}=<n> (CONC.SCALE.{Y|X|Z}): Concentration scale factor is valid only if concentration function IS a 1D profile. This allows the rolloff to be shortened (value <1.0) or lengthened (value >1.0) using this linear scalar factor. The default value is 1.0, of coarse.

COLOR=<n>: Color used to display region during DEVEDIT in X color windows mode. This is an RGB bitmap with eight bits colper color. DEVEDIT has only a limited subset of these colors so the closest match is used. Some basic colors can be specified by name; such as, red, green, blue, yellow, cyan, magenta, black, and white. See Generic Parameters for a more complete description.

Replaces Card

```
AddImpurity      [ID=<n>]      [Region=<n>]      [Color=<n>]      Base1=<point2d>
Base2=<point2d> [PeakValue=<n>] \ [ReferenceValue=<n> | ContourValue=<n>] \
[{Z1=<n> Z2=<n>} | Z=<n>,<n>] \ CombinationFunction=<c> \
Rolloff1=<c> ConcentrationFunction1=<c> \ Coefficient1=<n> | CParam1=<n>
[CRatio1=<n>] \ Rolloff2=<c> ConcentrationFunction2=<c> \
Coefficient2=<n> | CParam2=<n> [CRatio2=<n>] \
Rolloff3=<c> ConcentrationFunction3=<c> \ Coefficient3=<n> | CParam3=<n>
[CRatio3=<n>]
```

See Also

PROFILE

9.13.10: IMPURITY REFINEMENT

Set limit on the impurity differential across triangles.

Syntax

```
IMPURITY.REFINE IMPURITY=<C> [ID=<N>] SENSITIVITY=<N> \
[scale=<C>] [transition=<N>]
```

or

```
IMPURITY.REFINE MINIMUM.SPACING=<N> | Z=<N>
```

or

```
IMPURITY.REFINE ID=<N> DELETE
```

Preferred Abbreviation: `imp.ref`

Description

These values are used by the meshing routine to determine if triangles are small enough. When the mesh card is run, the current sensitivity of each impurity is tested. If the impurity difference across the triangle is greater than sensitivity, the triangle is broken into smaller triangles.

Parameters

ID=<n> - Used to delete or modify an exist refinement. If no id is supplied, the first unused id (starting from 1) is used.

DELETE - Deletes the impurity refinement identified by parameter id.

IMPURITY=<c> - Specifies what quantity/impurity is being refined. (i.e. Boron, Arsenic, Potential, etc...). See Generic Parameters for a more complete description.

SENSITIVITY=<n> If an impurity's value changes more than sensitivity, smaller triangles are created. If an impurity's scale is logarithmic, sensitivity is in powers of ten. Impurities are really extrapolated using arc hyperbolic sine (not using logarithms) and then normalized. There should be no noticeable difference on values greater than 10 times the transition value (one level of sensitivity).

SCALE=<C> - Specifies which scale the sensitivity should use. Different impurities (quantities) have different default scales. See Generic Parameters - Impurity for a default values

Linear - use linear scale

Logarithmic (log) - an alias for arc.hyperbolic.sine

arc.hyperbolic.sine (arc.h.sine) - use arc hyperbolic sine scale which is similar to a logarithmic scale.

TRANSITION=<N> This value is used to modify the arc.hyperbolic.sine (log) scale. Values below this value are considered insignificant. Different impurities (quantities) have different default transition values. See Generic Parameters - Impurity for default values.

MINIMUM.SPACING=<n> (MIN.SPAC) If a triangle is narrower than this, it is not narrowed further. If a triangle is shorter than this, it is not shortened. This parameter applies to all impurities currently being refined, not just the impurity specified by this card.

3-D Parameters

Z=<n> Refine at the specified z plane

Replaces Card

```
ImpRefine[Type=<c> Value=<n> | Sensitivity=<n>] MinSpacing=<n> Z=<n>
```

See Also

MESH

9.13.11: INITIALIZE

Clear existing device and load file.

Syntax

```
INIT INFILE=<C> [ Z=<POINT2D> | { Z1=<N> Z2=<N> } ] MESH[ =<BOOLEAN> ]  
LOAD FILE.NAME=<C> [ [ TYPE= ] <C> ] [ Z=<POINT2D> | { Z1=<N> Z2=<N> } ]  
MESH[ =<BOOLEAN> ]
```

Parameters

FILE.NAME=<c> (**file**, **infile**, **inf**) File name of a Silvaco standard structure file, or a DEVEDIT command file.

TYPE=<c> Override the automatic file type recognition and load file as the specified type.

SILVACO standard(mas) = Silvaco standard structure file.

structure(str) = Silvaco standard structure file.

card.deck(deck) = DEVEDIT command file.

MESH[=<boolean>] If mesh is set to false, any mesh commands are ignored and structure files are loaded without their mesh. This can greatly speed up load time. The old mesh is not needed if you are just trying to remesh a device. The default is mesh=true (accept mesh commands and load structure file meshes).

3-D Parameters

Z1=<n> If a 2D region is loaded, convert to a 3D region, using z1 as the starting z plane.

Z2=<n> If a 2D region is loaded, convert to a 3D region, using z2 as the ending z plane.

Z=<n>,<n> z1,z2 as one parameter.

Replaces Card

```
LoadFile FileName=<c> [Type=<c>] [Z=<point2d> | {Z1=<n> Z2=<n>}]
```

See Also

STRUCTURE

9.13.12: JOIN

Join two devices together.

Syntax

```
JOIN[ [SIDE=]RIGHT|LEFT|TOP|BOTTOM] FILE.NAME=<C> \
[ADJUST=<N>|SURFACE.ALIGN[=<BOOLEAN>]] [SPACER.THICKNESS=<N> \
[SPACER.MATERIAL=<C>]] [MIRROR[=<BOOLEAN>] [AUTOMATIC.JOIN[=<BOOLEAN>]]]
```

Description

Combine the device currently loaded in DEVEDIT and a device stored in a file into one device.

Parameters

[SIDE=]RIGHT|LEFT|TOP|BOTTOM Side on which to join the new structure onto. The default is side=right.

FILE.NAME=<c> Name of the file containing the device to be joined in DEVEDIT command format or Silvaco standard structure file format.

ADJUST=<n> If side=right or side=left, adjust the device down (or up if negative) a specified amount before performing join. If side=top or side=bottom, adjust the device to the right (or to the left if negative) the specified amount before performing join.

SURFACE.ALIGN[=<boolean>] If true perform the necessary adjust **surf.align** to have the semiconductor regions in both devices aligned at the top or left side of the joined device. The default is **surface.align=false**.

SPACER.MATERIAL=<c> (sp.mat) Specifies which material joins the two devices (i.e., Silicon, Aluminum, AlGaAr, etc...). See Generic Parameters for a more complete list.

SPACER.THICKNESS=<n> The thickness of the spacer inserted between the two devices.

MIRROR[=<boolean>] (FLIP) Take a mirror image of the new device before merging it into the existing device. The default is not to mirror the new device.

AUTOMATIC.JOIN[=<boolean>] (AUTO.JOIN) If two regions made up of the same material presently touch, the regions will be joined into one region. The attributes from the region with the lowest id is used for the combined region. The default value is true.

Note: As of this edition, the spacer parameters were not ready for release and can not be included in Version 2.0.0.

9.13.13: MESH

Create new mesh using previous set parameters.

Syntax

```
MESH[ [MODE=]MESH.BUILD|TENSOR.PRODUCT|DELETE ] ]
```

Description

When the mesh card is used, the following steps are performed.

Deletes any existing mesh.

1. Performs automatic boundary conditioning (if set). See **bounday.conditioning** card for more details.
2. Creates a base mesh. In **mesh.build** mode, creates a tensor product mesh using **base.mesh** parameters. See **base.mesh** card for more details. In **tensor.product** mode, it creates a tensor product mesh using all currently existing boundary points, those remaining after boundary conditioning.
3. Refines on geometry. Any points not part of the base mesh are now handled.
4. Refines on impurities. See **impurities.refine** card.
5. Refine on mesh constrains. See **constrain.mesh** card.

Parameters

```
[mode=]mesh.build|tensor.product
```

Examples

```
# build mesh using mesh build algorithm.
mesh
mesh mode=mesh.build
mesh mesh.build

# build a first level tensor product mesh.
mesh tensor.prod

# delete existing mesh
mesh del
```

See Also

```
BASE.MESH, BOUNDARY.CONDITIONING,
IMPURITY.REFINE, CONSTRAIN.MESH
```

9.13.14: MIRROR

Mirror the device.

Syntax

```
MIRROR [ [SIDE=]RIGHT|LEFT|TOP|BOTTOM] [AUTOMATIC.SPLIT[=<BOOLEAN>]]
```

Description

Mirror the device by creating a mirror image of the device in the given direction and joining the image to that side.

Parameters

[SIDE=]RIGHT|LEFT|TOP|BOTTOM Direction to mirror the device. The default direction is right.

AUTOMATIC.SPLIT[=<boolean>] (**AUTO.SPLIT**) If the region does not touch the mirrored edge, the region will be split into two discrete regions. If **auto.split=false**, the region is disjointed, but remains one region. The default is **auto.split=true**.

9.13.15: MOVE

Move the device around in the work area.

Syntax

```
MOVE { [DIRECTION=] { RIGHT | LEFT | UP | DOWN } DISTANCE=<N> } \
| | { [X.ADJUST=<N>] [Y.ADJUST=<N>] }
```

Description

Move the device around in the work area. Does not change the device at all, only its relative position in space.

Parameters

[DIRECTION=]{RIGHT|LEFT|UP|DOWN} (DIR) Direction to move the device. If this parameter is used, a distance parameter must also be supplied. The default direction is right.

DISTANCE=<n> (DIST) The distance (in microns) to move the device in the specified direction.

X.ADJUST=<n> (X.ADJ) Move the x coordinates of all point by adding x.adjust. x.adjust may be negative.

Y.ADJUST=<n> (Y.ADJ) Move the y coordinates of all point by adding y.adjust. y.adjust may be negative.

Examples

```
move dir=right dist=3
# point 0,0 becomes 3,0

move left dist=3
# point 0,0 becomes -3,0

move y.adj=3
# point 0,0 becomes 0,3

move x.adj=-3 y.adj=3
# point 0,0 becomes -3,3
```

9.13.16: PROFILE

Add, replace, or delete a 1D profile.

Syntax

```
ROFILE {NAME=<C> DELETE} \
| {[NAME=<C>] IMPURITY=<C> FILE.NAME=<C> | INFILE=<C> | DATA.POINT=<POINT_2D>}
```

Description

Set up a 1-D doping profile that can later be used to distribute an impurity using the impurity card.

Parameters

NAME=<c> Name is used to identify a 1D profile. This field is required in delete mode. If no name is given and delete is not specified, a new name in the form “NewProfile000” is used. There are a limited set of names that should not be used. See the impurity card for more details.

DELETE Delete the profile identified by the name parameter.

IMPURITY=<c> Specified which impurity this profile is **describing.imp** (i.e., Boron, Arsenic, Potential, etc...). See Generic Parameters for a more complete description.

FILE.NAME=<c> File name of a SUPREM3 or a 1-D SUPREM4 Silvaco Standard Structure file. The selected impurity is extracted in file and the distance from the surface of the semiconductor inf and the value at that location are stored. Only the data points in the semiconductor are stored.

DATA. POINT=<POINT_2D> Each data point consists of a distance from d.p the peak value (surface) and the value (concentration) at that location.

Examples

```
# Load an arsenic profile from a suprem3 file
profile file=suprem3.str name="Arsenic Profile" imp=arsenic

# Make a phosphorous profile
profile name=PhosProfile imp=phosphorous d.p=0,7e19 d.p=0.2,1e20 \
d.p=1,1e19 d.p=2,1e18 d.p=3,1e17

# delete the arsenic
profile delete name="Arsenic Profile"
```

Replaces Card

```
Profile [Name=<c>] Impurity=<impurity> \ FileName=<c> |
DataPoint=<point_2d>...
```

See Also

IMPURITY

9.13.17: QUIT

Exit DEVEDIT or end reading file.

Syntax

```
BYE,END,EXIT,QUIT
```

Description

Any one of these four cards cause DEVEDIT to exit or, if reading a file, the remainder of the file is ignored.

9.13.18: REFINE

Manually refine existing mesh.

Syntax

```
REFINE [DIRECTION=]X|Y|BOTH|UNREFINE \
{X1=<N> Y1=<N> X2=<N> Y2=<N>} \
| {POINT.1=<N>,<N> POINT.2=<N>,<N>} \
| {LEFT=<N> TOP=<N> RIGHT=<N> BOTTOM=<N>}
```

Description

More (less if unrefine) triangles are created in the x, y, or both directions in the specified rectangle. A mesh must currently exist that was not loaded from a structure file.

Parameters

[DIRECTION=]x|y|both|unrefine (DIR) Refine direction.

x = more triangles horizontally. (~twice as many)

y = more triangles vertically. (~twice as many)

both = more triangles. (~four times as many)

unrefine = less triangles.

X1=<n> (LEFT) Left side of box to (un)refine in microns.

Y1=<n> (TOP) Top of box to (un)refine in microns.

X2=<n> (RIGHT) Right side of box to (un)refine in microns.

Y2=<n> (BOTTOM) Bottom side of box to (un)refine in microns.

POINT.1=<n>,<n>X1,Y1 (P1) as one parameter.

POINT.2=<n>,<n>X2,Y2 (P2) as one parameter.

Replaces Existing Card

```
Refine Mode=<C> P1=<POINT2D> P2=<POINT2D>
```

See Also

MESH

9.13.19: REGION

Add, replace or delete a region.

Syntax

```
REGION {DELETE {ID=<n>|ID=<c>|NAME=STRING} \
| {{ID=<n>}[NAME=<c>]}|ID=<c>} [MATERIAL=<c>] \
[COLOR=<n>] [PATTERN=<n>] POINTS=<point2d_list> \
[WORK.FUNCTION=<n>] [ELECTRODE.ID[=<n>]] [Z1=<n>] [Z2=<n>]}
```

Preferred Abbreviation

REG

Description

A DEVEDIT device is made up of regions. Each region has a unique region id number. Regions also can have a name. Several regions can have the same name. Region names should not start with a number. Some places accept a region id number or a region name and will assume a number is a region id number.

Regions are made of a material and contain a list of non-intersecting polygons. A region can be made up of several discrete polygons. If one polygon is contained in another polygon, it is considered a hole in the containing polygon. If a polygon is inside a “hole” polygon, it describes a region with material, a hole in the hole.

For display purposes (inside DEVEDIT only), a region can contain a color and a fill pattern. This is only used when DEVEDIT opens an X display window.

A region can also be an electrode, in that case the electrode id number can be supplied or one can be chosen by DEVEDIT. For future enhancement to simulators, a work function can also be supplied.

Parameters

DELETE Used to delete the region identified by **id=** or **name=**. One and only one parameter (**id=** or **name=**) should be used. If id is a number, delete region with that number. If id is a string, delete the region name with the lowest region number. If **name=** is used, delete all region names.

ID=<n> A region number that uniquely identifies the region to be inserted, replaced or deleted.

ID=<c> Identifies the region with name <c> to be replaced or deleted. If no region has name <c>, insert a new region with name <c> and the lowest unused region number. If two regions have name <c>, the region with the lowest region number will be replaced or deleted.

NAME=<c> Deletes all regions with name <c> when in delete mode. Otherwise, the region being replaced or inserted will be given the name <c>.

MATERIAL=<c> Specifies what material the region is made of (i.e., Silicon, Aluminum, AlGaAs, etc...). See Generic Parameters for a more complete description.

COLOR=<n> (COLOUR) Used to display region during DEVEDIT in X windows mode. This is an RGB bitmap with eight bits per color. DEVEDIT has only a limited subset of these colors, therefore the closest match is used. Some basic colors can be specified by name; such as, red, green, blue, yellow, cyan, magenta, black, and white. See Generic Parameters for a more complete description.

PATTERN=<n> Used as fill pattern for display region in DEVEDIT in X windows mode. See Generic Parameters for a more complete description.

POINTS=<point_2d_list> Used for location of points making polygons that describe the region.

ELECTRODE.ID[=<n>] (ELEC.ID) Describes the region as an electrode, setting the electrode number to <n>. If <n> is not supplied, the lowest unused electrode id number will be used.

WORK.FUNCTION=<n> Used only if electrode.id is set to define work function for materials. This is not currently being used by any simulators, but may be used in future releases.

3D Parameters

Z1=<n> is the Z plane where region starts (in microns) for the 3D Mode.

Z2=<n> is the z plane where region ends (in microns) for the 3D Mode.

Examples

```
# ADD A 10 BY 10 MICRON BLOCK OF SILICON AS REGION #1
# WITH THE NAME "WAFFER" DISPLAYED IN RED
REGION ID=1 NAME=WAFFER MATERIAL=SILICON COLOR=0X30 \
POINTS="0,0 10,0 10,10 0,10 0,0"

# ADD AN ALUMINUM CONTACT AS ELECTRODE #1 USING THE NEXT
# AVAILABLE REGION NUMBER WITH THE NAME "SOURCE"
# USING DEFAULT COLOR.
REGION NAME=SOURCE MATERIAL=AL ELEC.ID=1 WORK.FUNC=4.28 \
POINTS="0,-1 1,-1 1,0 0,0 0,-1"

# SET REGION #3 TO ELECTRODE #2
REGION NAME=SOURCE MATERIAL=AL ELEC WORK.FUNC=4.28 \
POINTS="9,-1 10,-1 10,0 9,0 9,-1"

# DELETE REGION #2
REGION DELETE ID=2

# DELETE THE FIRST REGION NAMED SOURCE
REGION DELETE ID=SOURCE

# DELETE ALL REGIONS NAMED SOURCE
REGION DELETE NAME=SOURCE
```

Replaces Cards

```
AddRegion [Region=<n>][Name=<c>][Electrode=<n>] \
[Material=<c>][Color=<n>][Pattern=<n>] \
Points=<point2d_list> [WorkFunction=<n>][Z1=<n>][Z2=<n>]
DeleteRegion Region=<n>
```

9.13.20: RENUMBER.REGIONS

Renumber all regions to fill in gaps or specifically renumber one region.

Syntax

```
RENUMBER.REGIONS [FROM=<n> TO=<n>]
```

Parameters

FROM=<n> Region ID of region to be renumbered.

TO=<n> Region's new Region ID.

9.13.21: SOURCE

Run commands stored in file (on existing device).

Syntax

```
SOURCE FILE.NAME=<C> [Z1=<N> Z2=<N>] MESH[=<BOOLEAN>]
```

Parameters

FILE.NAME (FILE) Run DEVEDIT cards contained in the specified file.

MESH=<boolean> If mesh is set to false, any mesh commands are ignored. The default is mesh=true (accept mesh commands).

3-D Parameters

Z1=<N> If a 2D region card is read, convert to a 3D region, using z1 as the starting z plane.

Z2=<N> If a 2D region card is read, convert to a 3D region, using z2 as the ending z plane.

Replaces Card

```
IncludeFile FileName=<c> [Type=<c>][Z=<point2d>|{Z1=<n> Z2=<n>}]
```

9.13.22: STRETCH

Stretch the device.

Syntax

```
STRETCH{X.VAL=<N>|Y.VAL=<N> STRETCH.VALUE=<N>}  
| {[REGION.ID=<C> | MATERIAL=<C>] LENGTH=<N>\  
| STRETCH.VALUE=<N>Y.LENGTH=<N>|Y.STRETCH.VALUE=<N> \  
CENTER[=<BOOLEAN>]} \  
| {X1=<N>X2=<N>STRETCH.VALUE=<N>|LENGTH=<N>} \  
| {Y1=<N>Y2=<N>STRETCH.VALUE=<N>|LENGTH=<N>}
```

Description

Stretch provides many ways to make the device longer or taller or even narrower or shorter. There are many parameters that can be used in many combinations, however, most of these parameters are only for the most advanced user. Most users should be able to look at the first five (5) examples below and learn all they need about stretch.

Stretch allows a vertical or horizontal line to be stretched out a specified amount or allows a range to be evenly stretched.

The line to be stretched can be a specific location ($x=1$). Alternatively, the line can be the center of a region, specified by the region's name, id or material. If a line is to be stretched into an area, the width of the new area must be supplied. A `stretch.value` can be supplied to specify a new length. Alternatively, a new length can be determined for the specified region. That is, the amount stretched equals the desired length minus the original length of the region.

To stretch an area to cover an even larger area, an area can be specified by region name, region id, material name, $x1/x2$ pair, or an $y1/y2$ pair. If a region is specified, the center option must be set to false. This area can be given a new length (using `length`) or an extra amount (using `stretch.value`) to be added to the length.

Length and `stretch.value` are normally assumed to apply to the x direction, unless `y.val` or $y1/y2$ are used. `y.length` or `y.stretch.value` can be used to force the stretch to be in the y direction. Mixed parameters like `x.val` and `y.stretch.value` should not be used together.

In general, it is only useful to do line type stretches in areas where the impurities are fairly constant, like the center of the gate on a MOSFET transistor. This allows a gate's length to be quickly changed for multiple simulations. This is described in the first five(5) examples below.

Parameters

LENGTH=<n> (LEN, X.LENGTH, X.LEN) This is the new x length of the specified region, material, $x1/x2$ area, or $y1/y2$ area. If none of these are specified, the first region made of poly silicon is used. If no such region exists, it is an error. Care should be taken not to shrink a region. A warning is issued if the region shrinks. If `y.val` or $y1/y2$ are used, this is the same as `y.length`.

Y.LENGTH=<n> (Y.LEN) This is the new y length of the specified region, material, or $y1/y2$ area. If none of these are specified, the first region made of poly silicon is used. If no such region exists, it is an error. Care should be taken not to shrink a region. A warning is issued if the region shrinks.

REGION.ID=<c> This is the name or id number of the region used to identify the stretch line at the center of this region. If `center=false`, then the whole region is stretched.

MATERIAL=<c> The first region found made of this material is used to identify the stretch line at the center of that region. If `center=false`, then the whole region is stretched. See Generic Parameters for a list of materials.

X.VAL=<n> (X) Stretch from the vertical line $x=x.val$. `stretch.value` must be supplied with this parameter.

Y.VAL=<n> (Y) Stretch from the vertical line $x=x.val$. `stretch.value` must be supplied with this parameter.

STRETCH.VALUE=<n> (STR.VAL, X.STR,VAL) The device is stretched this much longer. If the `x.val` or `y.val` parameters are used, the stretch expands that line to the new width. If $x1$ and $x2$ or $y1$ and $y2$ parameters are used, the stretch is distributed throughout the range given.

Y.STRETCH.VALUE=<n> (Y.STR.VAL) Same as `stretch.value` except stretch in the y direction. If `y.val` or $y1/y2$ are use `stretch.value` is also in the y direction.

CENTER[=<boolean>] This cause any stretch to happened at the center of the specified region. `center=false` causes the stretch to be distributed over the region. Center is defaulted to true unless $x1/x2$ or $y1/y2$ parameters are used, in which case, it defaults to false.

X1=<n> Start of x direction stretch. Must use stretch.value or length with this parameter.

X2=<n> End of x direction stretch. Must use stretch.value or length with this parameter.

Y1=<n> Start of y direction stretch. Must use stretch.value or y.length with this parameter.

Y2=<n> End of y direction stretch. Must use stretch.value or y.length with this parameter.

Examples

```
# cause the device to be stretched at the center of first
# polysilicon region so that the new length of that region
# will be 1.5 microns in the x direction.
stretch length=1.5
```

```
# cause the device to be stretched at the center of first
# polysilicon region so that the new length of that region
# will be an extra 1.5 microns in the x direction.
stretch stretch.value=1.5
```

```
# cause the device to be stretched at the center of first
# polysilicon region so that the new length of that region
# will be 1.5 microns in the x direction.
stretch material=polysilicon length=1.5
```

```
# cause the device to be stretched at the center of region #3
# so that the new length of region #3 will be 1.5 microns in the # x
# direction.
stretch reg=3 length=1.5
```

```
# cause the device to be stretched at the center of the region
# named "gate" so that the new length of that region will be
# 1.5 microns in the x direction.
stretch reg=gate length=1.5
```

```
# cause the device to be stretch in the x direction so that new
# length of the region named "gate" will be 1.5 microns. The
# extra area will be evenly distribute across the whole gate.*
stretch reg=gate ^center length=1.5
```



```
# cause the region named "gate" to be stretch to 1.5 microns
# in the y direction. The device is expanded at the center of the gate.
stretch reg=gate y.length=1.5

# add an extra.5 microns in depth at y=2.0
stretch y.val=2.0 stretch.value=.5

# stretch out the doping profile between 2 microns in depth
# and 3 microns in depth to take 3 microns instead of 1 micron.
stretch y1=2.0 y2=3.0 length=3.0
```

Note: ^center is the same as center=false as described in generic parameters - boolean type.

9.13.23: STRUCTURE

Save current structure to a file.

Syntax

```
STRUCTURE OUTFILE=<C> # TYPE=MASTER (DEFAULT)
SAVE FILE.NAME=<C> [TYPE=<C>] \ # TYPE=CARD.DECK (DEFAULT)
[ [TYPE=]MASTER|STRUCTURE|CARD.DECK]
```

Parameters

FILE.NAME=<c> (FILE, OUTFILE, OUTF) File name used to store a Silvaco Standard Structure file or a DEVEDIT command file. The structure card only stores Silvaco Standard Structure files.

TYPE=<c> Type of file to store. Possible values are:

- **MASTER(MAS)** = **SILVACO** standard structure file.
- **STRUCTURE(STR)** = **SILVACO** standard structure file.
- **CARD.DECK(DECK)** = **DevEdit** command file.

3-D Parameters

TYPE=<c> In 3D mode, structure files are normally saved with prismatic elements. mode=tetrahedrons (or mode=tet) can be used to output tetrahedral elements into a structure file.

Replaces Card

```
SaveFile FileName=<c> [Type=<c>]
```

9.13.24: SUBSTRATE

Special substrate electrode.

Syntax

```
SUBSTRATE DELETE \  
| { [NAME=<C>] [ELECTRODE.ID=<N>] [WORK.FUNCTION=<N>] [APPLY[=<BOOLEAN>]] }
```

Preferred Abbreviation

SUBSTR

Description

The substrate is a special region with no thickness that is placed at the bottom of the device. The substrate is always considered to be an electrode.

Parameters

DELETE Substrate not longer exists.

NAME=<c> The name of the substrate electrode. (default=substrate)

ELECTRODE.ID=<n> (elec.id) Describes region as an electrode setting the electrode elec number to <n>. If <n> is not supplied, the lowest used electrode id number will be used.

WORK.FUNCTION=<n> (**work.func**) Used only if electrode.id is used to set work function for materials. This is not currently used by any simulators, however may be used in future releases.

APPLY[=<boolean>] (for internal use only; default=true) If apply=false, only set parameter in the substrate panel.

Examples

```
# Make the substrate electrode #3 and named collector.  
substr name=collector elec.id=3 work.func=4.28
```

Replaces Card

```
Substrate Delete \  
| { [Name=<c>] [Electrode=<n>] \  
  [WorkFunction=<n>] [NoApply[=<boolean>]] }
```

9.13.25: WORK.AREA

Set viewing area of canvas.

Syntax

```
WORK.AREA{ [X1=<N>] [X2=<N>] [Y1=<N>] [Y2=<N>] } \  
{ [LEFT=<N>] [RIGHT=<N>] [TOP=<N>] [BOTTOM=<N>] } \  
| { [POINT.1=<POINT2D>] [POINT.2=<POINT2D>] }
```

Description

Set the viewing area seen in the main window of DEVEDIT (X windows mode only).

Parameters

X1=<n> (LEFT) Minimum x value of draw area.
X2=<n> (RIGHT) Maximum x value of draw area.
Y1=<n> (TOP) Minimum y value of draw area.
Y2=<n> (BOTTOM) Maximum y value of draw area.
POINT.1=<n>,<n> (P1) $x1,y1$ as a single parameter
POINT.2=<n>,<n> (P2) $x2,y2$ as a single parameter

Replaces Card

```
WorkArea P1=<point2d> P2=<point2d>
```

9.13.26: Z.PLANES

Define z planes (3-D mode only).

Syntax

```
Z . PLANE [ { Z=<n> DELETE } | Z=<n> [ SPACING=<n> ] ] [ MAXIMUM.SPACING=<n> ] \
[ MAXIMUM.RATIO=<n> ]
```

Description

Z planes are created at the front and back of all regions and at the specified z locations. Additional z planes are added to meet the requirements of spacing, maximum.spacing, and maximum.ratio.

Parameters

Z=<n> Z plane that must exist or is to be deleted.
DELETE Delete the specified z plane and any associated spacing.

Note: If a z plane at the start or end of a region is deleted, only the spacing associated with that z plane is deleted. The Z plane will still be made part of the structure.

SPACING (SPAC) The spacing around the specified z plane will be restricted to the specified spacing. spacing=0 means there are no restrictions specific to this z plane.spacing=0 is the default case.

MAXIMUM.SPACING=<n> (MAX.SPAC) This is the maximum gap (in microns) between two adjacent z planes.

MAXIMUM.RATIO=<n> (MAX.RATIO) This is the maximum ratio of the gap between two adjacent z planes and the gap to the next z plane.

Replaces Card

```
ZPlane Z=<n> [MaxSpacing=<n>][MaxRatio=<n>]
```

9.13.27: GENERIC PARAMETER - BOOLEAN TYPE

Parameters described like `param[=<boolean>]` are boolean parameters, meaning they can be either true or false.

There are several ways to set the parameter to true. They are:

- `param=true`
- `param #just` listing the param sets it to true
- `param=on`
- `param=1`
- `param=yes`

There are several ways to set the parameter to false. They are:

- `param=false`
- `! param`
- `^ param`
- `param=off`
- `param=0`
- `param=no`

The default value for a boolean parameter type is dependent on the specific parameter.

9.13.28: GENERIC PARAMETER - COLOR

In X window mode, regions and objects can have a color assigned to them. Regions have a default color base on their material. However, the card with color parameters can override these colors.

The color can be set using a standard 24 bit number with eight bits for each color component, red, green, and blue. The easiest way to use this is use hexadecimal values by starting the number with "0x". Each component than has a range from 00 (no color) to FF (full color). See the example below.

```
color=0xFF0000# Full red
```

```
color=0x00FF00# Full green
```

```
color=0x0000FF# Full blue
```

Not all color combinations are supported by DEVEDIT. The closest color support by DEVEDIT is used. The eight primary colors can be set by name (i.e., `color=red`).

Color Name	Color Value
black	0x000000
red	0xFF0000
green	0x00FF00
blue	0x0000FF
cyan	0x00FFFF
yellow	0xFFFF00
magenta	0xFF00FF
white	0xFFFFFFFF

In hexadecimal, digits are: 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F.

Note: 0x00 < 0x09 < 0x0A < 0x0F < 0x10 < 0xA0 < 0xF0 < 0xFF.

Each component should be considered separately.

9.13.29: GENERIC PARAMETER - IMPURITY

Any of the following names can be used for impurity parameters. Names can be abbreviated as long as the individual words remain unique in the list. A dot(.) can be used as a word separator, otherwise multiple word impurities must be quoted. The short name is the preferred abbreviation. Special effort will be made to keep these names unique when adding new impurities.

Note: The number symbol(#), the equal sign(=), the single quote (') and the space symbol () must be quoted.

Possible Values

Impurity Full Name	Preferred Abbreviations	Default imp.refine	
		Scale	Transition Value
"Vacancies"	vac	log	1.0
"Interstitials"		log	1.0
"Arsenic"		log	1.0e+10
"Phosphorus"	phos	log	1.0e+10
"Antimony"		log	1.0e+10
"Boron"		log	1.0e+10
"Donors"		log	1.0e+10
"Acceptors"		log	1.0e+10
"Composition Fraction X"*	comp.fract.x	linear	
"Composition Fraction Y"*	comp.fract.y	linear	
"Electron Conc Process Simulation"*		log	1.0
"Hole Conc Process Simulation"*		log	1.0
"X Velocity Process Simulation"*	x.vel.p.s	linear	
"Y Velocity Process Simulation"*	y.vel.p.s	linear	
"Dry O2"*		log	1.0
"Wet O2"*		log	1.0
"Interstitial Traps"*		log	1.0

Impurity Full Name	Preferred Abbreviations	Default imp.refine	
		Scale	Transition Value
"Gold Conc"*	gold	log	1.0
"Cesium"		log	1.0
"Delta Area"*		linear	
"Stress XX"*		linear	
"Stress XY"*		linear	
"Stress YY"*		linear	
"Fixed Oxide Charge"*		linear	
"Potential"	pot	linear	
"Device Potential"*	dev.pot	linear	
"N Mobility"	n.mob	log	1.0
"P Mobility"	p.mob	log	1.0
"Total Field"*		log	1.0
"Fixed Charge"*		log	1.0
"Impact Ionization Rate"*		linear	
"N Carrier Conc"*	n.car	log	1.0
"P Carrier Conc"*	p.car	log	1.0
"Conduction Current"*	cond.cur	linear	
"Displacement Current"*	disp.cur	linear	
"Total Current"*	t.cur	linear	
"Electron QFL"*		linear	
"Hole QFL"*		linear	
"Valency Band Potential"*	val.band.pot	linear	
"Conduction Band Potential"*	cond.band.pot	linear	
"NetDoping" "Net Doping"*	net.dop	log	1.0e+10
"AbsNetDoping" "Abs Net Doping"*	abs.net.dop	log	1.0e+10
"Charge Concentration"*		log	1.0
"Carrier Concentration"*	car	log	1.0
"Recombination Rate"*		log	1.0

Impurity Full Name	Preferred Abbreviations	Default imp.refine	
		Scale	Transition Value
"Impact Gen'd Carriers"*	impact.g.car	log	1.0
"X Dir Electric Field"*		linear	
"Y Dir Electric Field"*		linear	
"Z Dir Electric Field"*		linear	
"Hole Temperature"*		linear	
"Elec Temperature"*		linear	
"Semiconductor temp"*		linear	
"Hole temp gradient"*		linear	
"Electron temp grad"*		linear	
"Electron velocity"*	elec.vel	linear	
"Hole velocity"*	hole.vel	linear	
"Intrinsic Conc (nio)"*		log	1.0
"Aluminium conc"*		log	1.0e+10
"Indium conc"*		log	1.0e+10
"Gallium conc"*		log	1.0e+10
"Carbon conc"*		log	1.0
"QFL Gradient X-comp"*		linear	
"QFL Gradient Y-comp"*		linear	
"Total Field 2"*		linear	
"PAC"		linear	
"Intensity"		linear	
"Norm Intensity"*		linear	
"Norm grad Int"*		linear	
"Total Doping"*		log	1.0e+10
"Net Active Doping"*		log	1.0e+10
"Active Boron"*	act.boron	log	1.0e+10
"Active Phosph"*	act.phos	log	1.0e+10
"Active Arsenic"*	act.arsen	log	1.0e+10
"Active Antimony"*	act.antim	log	1.0e+10
"Insulator Charge"*	ins.ch	log	1.0

Impurity Full Name	Preferred Abbreviations	Default imp.refine	
		Scale	Transition Value
"Interface Charge"*	int.ch	log	1.0
"Semi Fixed Charge"*		log	1.0
"Slow State Density"*		log	1.0
"Fast State Density"*		log	1.0
"Occupancy Trap 1"*	occ.trap.1	linear	
"Occupancy Trap 2"*	occ.trap.2	linear	
"Occupancy Trap 3"*	occ.trap.3	linear	
"Occupancy Trap 4"*	occ.trap.4	linear	
"Occupancy Trap 5"*	occ.trap.5	linear	
"Occupancy Trap 6"*	occ.trap.6	linear	
"Occupancy Trap 7"*	occ.trap.7	linear	
"Occupancy Trap 8"*	occ.trap.8	linear	
"Occupancy Trap 9"*	occ.trap.9	linear	
"Occupancy Trap 10"*	occ.trap.10	linear	
"Electron Current X-comp."*	elec.cur.x	log	1.0
"Electron Current Y-comp."*	elec.cur.y	log	1.0
"Electron Current Z-comp."*	elec.cur.z	log	1.0
"Hole Current X-comp."*	hole.cur.x	log	1.0
"Hole Current y-comp."*	hole.cur.y	log	1.0
"Hole Current Z-comp."*	hole.cur.z	log	1.0
"Current X-component"*	cur.x	log	1.0
"Current Y-component"*	cur.y	log	1.0
"Current Z-component"*	cur.z	log	1.0
"Displ Current X-component"*	disp.cur.x	log	1.0
"Displ Current Y-component"*	disp.cur.y	log	1.0
"Photo Generation rate"*		log	1.0
"Hole Current magnitude"*	hole.cur.mag	log	1.0
"Electron Current magnitude"*	elec.cur.mag	linear	
"Current Magnitude"*	cur.mag	linear	
"Current Flow"*	cur.flow	linear	

Impurity Full Name	Preferred Abbreviations	Default imp.refine	
		Scale	Transition Value
"Total Current (area)"*	t.cur.area	linear	
"Elec Cur Dens X"*	elec.cur.dens.x	linear	
"Hole Cur Dens X"*	hole.cur.dens.x	linear	
"Current Dens X"*	cur.dens.x	linear	
"Elec Cur Dens Y"*	elec.cur.dens.y	linear	
"Hole Cur Dens Y"*	hole.cur.dens.y	linear	
"N int.x"		linear	
"N int.y"		linear	
"P int.x"		linear	
"P int.y"		linear	
"Extended def size"*		linear	
"Extended def dens"*		linear	
"Current Dens Y"*	cur.dens.y	linear	
"Hole Mobility Lateral"*	hole.mob.lat	linear	
"Elec Mobility Lateral"*	elec.mob.lat	linear	
"Hole Mobility Transverse"*	hole.mob.trans	linear	
"Elec Mobility Transverse"*	elec.mob.trans	linear	
"Electron SRH rec. tno"*		linear	
"Hole SRH rec. tno"*		linear	
"Cooling package Temp"*		linear	
"Equilibrium potential"*		linear	
"Applied potential"*		linear	
"Joule Heat Power"*		linear	
"Total Heat Power"*		linear	
"Rec."*		linear	
"Heat conductivity"*		linear	
"Heat capacity"*		linear	
"Displacement J"*		linear	
"Displacement J X-comp"*		linear	
"Displacement J Y-comp"*		linear	

Impurity Full Name	Preferred Abbreviations	Default imp.refine	
		Scale	Transition Value
"Hole Diff coeff."*		linear	
"Electron Diff coeff."*		linear	
"Einstein Rel. Corr. Elect"*		linear	
"Einstein Rel. Corr. Holes"*		linear	
"Ionization Effect. field"*		linear	
"Ionization coeff h+"*		linear	
"Ionization coeff e-"*		log	1.0
"Total recombination rate"*		linear	
"SRH recombination rate"*		linear	
"Auger recombination rate"*		linear	
"User recombination rate"*		linear	
"Surface recombination rate"*		linear	
"Effect. min.Carr. lifetime"*	eff.min.car.lif e	linear	
"SRH rec. Effect. lifetime"*		linear	
"Auger Recom. Eff. lifetime"*		linear	
"User Recomb. Eff. lifetime"*		linear	
"Surf. Recom. Eff. lifetime"*		linear	
"Effective BGN"*		linear	
"Effective BGN from C-band"*		linear	
"Effective GN from V-band"*		linear	
"Effective nie"*		linear	
"Structure temp"*		linear	
"Heat capacitance"*		linear	
"Heat conductance"*		linear	
"Material density"*		linear	
"Package layer index"*		linear	
"Package material index"*		linear	
"Material Type #"*		linear	
"Sem./Ins Material #"*		linear	

Impurity Full Name	Preferred Abbreviations	Default imp.refine	
		Scale	Transition Value
"Region #"		linear	
"Mat.type # w/o electrodes"		linear	
"Sem./Ins # w/o electrodes"		linear	
"Region # w/o electrodes"		linear	
"Electrode #"		linear	
"SRH par. nl/nie"		linear	
"Relative Permittivity"		linear	
"Celsius Temp"		linear	
"D Vector Magnitude"		linear	
"D Vector X-comp"		linear	
"D Vector Y-comp"		linear	
"D Vector Z-comp"		linear	
"Cond. Current X-comp"	cond.cur.x	linear	
"Cond. Current Y-comp"	cond.cur.y	linear	
"Electron Velocity X-comp"	elec.vel.x	linear	
"Electron Velocity Y-comp"	elec.vel.y	linear	
"Hole Velocity X-comp"	hole.vel.x	linear	
"Hole Velocity Y-comp"	hole.vel.y	linear	
"Elec. Ionix. Eff. Field."		linear	
"Hole. Ionix. Eff. Field."		linear	
"Ratio nie/maj.carr.conc."		linear	
"Ratio nie/ni"		linear	
"Eff. BGN w/o T-depend."		linear	
"Thomson Heat Power"		linear	
"Net Doping (linear)"		linear	
"Total Doping (linear)"		linear	
"Charge Concentration (linear)"		linear	
"Heat Flow Density"		linear	
"X Dir Heat Flow Density"		linear	
"Y Dir Heat Flow Density"		linear	

Impurity Full Name	Preferred Abbreviations	Default imp.refine	
		Scale	Transition Value
"Z Dir Heat Flow Density"*		linear	
"dT/dx"		linear	
"dT/dy"		linear	
"dT/dz"		linear	
"Gradient of Time"*		linear	
"Z Plane Index"*		linear	

Note: The number symbol(#), the equal sign(=), the single quote (') and the space symbol () must be quoted.

9.13.30: GENERIC PARAMETER - MATERIAL

Any of the following names can be used for material parameters. Names can be abbreviated as long as the individual words remain unique in the list. A dot (.) can be used as a word separator, otherwise multiple word materials must be quoted. The short name is the preferred abbreviation. A special effort will be made to keep these names unique when adding new materials.

Note: The number symbol(#), the equal sign(=), the single quote (') and the space symbol () must be quoted.

Possible Values

Material Number (Structure File ID)	Material Full Name (TonyPlot Material Name)	Aliases (Compatability Names)
0	"Gas"	
1	"SiO2"	"Silicon Oxide" "SiO__2" "SiOxide" "Oxide"
2	"Si3N4"	"Silicon Nitride" "SiNO3" "SiNO__3" "SiNitride" "Nitride"
3	"Silicon"	"Si"
4	"Polysilicon"	"Poly"
5	"OxyNitride"	"OxyNit"
6	"Aluminum"	"Al"
7	"Photoresist"	"PhotoRes"

Material Number (Structure File ID)	Material Full Name (TonyPlot Material Name)	Aliases (Compatability Names)
8	"GaAs "	
9	"Sapphire "	
10	"Gold "	"Au "
11	"Silver "	"Ag "
12	"AlSi "	
13	"Tungsten "	"W "
14	"Titanium "	"Ti "
15	"Platinum "	"Pt "
16	"Palladium "	"Pd "
17	"Cobalt "	"Co "
18	"Molibdinum "	"Mo " "Md "
19	"Lead "	"Pb "
20	"Iron "	"Fe "
21	"Tantalum "	"Ta "
22	"AlSiTi "	
23	"AlSiCu "	
24	"AlGaAs "	
25	"InGaAs "	
26	"AlInAs "	
27	"InP "	
28	"Vacuum "	
29	"Fictive GaAs "	"GaAs-Hetro " "FGA "
30	"Mask Opaque "	
31	"Mask Clear "	
32	"Germanium "	"Ge "
33	"AlAs "	
34	"TEOS "	
35	"BSG "	
36	"BPSG "	
40	"Alpha Si 1 "	"alpha-Si#1 " "~a-Si__1 "
41	"Alpha Si 2 "	"alpha-Si#2 " "~a-Si__2 "

Material Number (Structure File ID)	Material Full Name (TonyPlot Material Name)	Aliases (Compatability Names)
42	"Alpha Si 3"	"alpha-Si#3" "~a-Si__3"
43	"Alpha Si 4"	"alpha-Si#4" "~a-Si__4"
50	"User #1"	"UD1(<user-defined-name>)"
...
59	"User #10"	"UD10(<user-defined-name>)"
60	"AlxGa1_xAs_x_0.25"	"AlxGa1-xAs, x=0.25" "ALG1" "AlGaAs1 (ALG1)"
61	"AlxGa1_xAs_x_0.5"	"AlxGa1-xAs, x=0.5" "ALG2" "AlGaAs2 (ALG2)"
62	"AlxGa1_xAs_x_0.75"	"AlxGa1-xAs, x=0.75" "ALG3" "AlGaAs3 (ALG3)"
63	"InxGa1_xAs_x_0.50 Unstr"	"InxGa1-xAs, x=0.50, unstrained" "ING0" "InGaAs (ING0)"
64	"InxGa1_xAs_x_0.33 Str GaAs"	"InxGa1-xAs, x=0.33, strained matched to GaAs" "ING1" "InGaAs (ING1)"
65	"InxGa1_xAs_x_0.75 Str InP"	"InxGa1-xAs, x=0.75, strained matched to InP" "ING2" "InGaAs (ING2)"
66	"AlxIn1_xAs_x_0.50"	"AlxIn1-xAs, x=0.50" "ALIN" "AlInAs (ALIN)"
69	"Barrier"	
70	"TiW"	
71	"PMMA"	
72	"SOG"	
73	"Polyimide"	
74	"Cooling package material"	
75	"Copper"	"Cu"
76	"Tin"	"Sn"
77	"Nickel"	"Ni"
78	"Ambient"	
79	"Air"	
80	"WSix"	"Tungsten Silicide" "WSi2"
81	"TiSix"	"Titanium Silicide" "TiSi2"
82	"NiSix"	"Nickel Silicide" "NiSi2"

Material Number (Structure File ID)	Material Full Name (TonyPlot Material Name)	Aliases (Compatability Names)
83	"CoSix"	"Cobalt Silicide" "CoSi2"
84	"TaSix"	"Tantulum Silicide" "TaSi2"
85	"PdSix"	"Paladium Silicide" "PdSi2"
86	"PtSix"	"Platinum Silicide" "PlSi2" "PtSi2"
87	"MoSix"	"Molybdenum Silicide" "MdSi2" "MoSi2" "Moly Silicide"
88	"ZrSix"	"Zirconium Silicide" "ZrSi2"
89	"AlSix"	"Aluminum Silicide" "AlSi2"
90	"Insulator"	
91	"Conductor"	
92	"Contact"	
99	"3C-SiC"	
100	"Diamond"	
101	"SiGe"	
102	"6H-SiC"	"a-SiC"
103	"4H-SiC"	"b-SiC"
104	"AlP"	
105	"AlSb"	
106	"GaSb"	
107	"GaP"	
108	"InSb"	
109	"InAs"	
110	"ZnS"	
111	"ZnSe"	
112	"ZnTe"	
113	"CdS"	
114	"CdSe"	
115	"CdTe"	
116	"HgS"	
117	"HgSe"	

Material Number (Structure File ID)	Material Full Name (TonyPlot Material Name)	Aliases (Compatability Names)
118	"HgTe "	
119	"PbS "	
120	"PbSe "	
121	"PbTe "	
122	"SnTe "	
123	"ScN "	
124	"GaN "	
125	"AlN "	
126	"InN "	
127	"BeTe "	
128	"InGaP "	
129	"GaSbP "	
130	"GaSbAs "	
131	"InAlAs "	
132	"InAsP "	
133	"GaAsP "	
134	"HgCdTe "	
135	"InGaAsP "	
136	"AlGaAsP "	
137	"AlGaAsSb "	
140	"SiN "	
141	"Computational Window"	
142	"Phase shift"	
143	"Si "	"Green's Silicon (for atlas) "
144	"Polymer "	
145	"CuInGaSe "	
146	"InGaN "	
147	"AlGaN "	
148	"InAlGaN "	
149	"InGaNAs "	

Material Number (Structure File ID)	Material Full Name (TonyPlot Material Name)	Aliases (Compatability Names)
150 ... 189	"User #11 " ... "User #50 "	"UD11 (<user-defined-name>) " ... "UD50 (<user-defined-name>) "
190	" InGaNP "	
191	"AlGaNaS "	
192	"AlGaNP "	
193	"AlInNaS "	
194	"AlInNP "	
195	" InAlGaAs "	
196	" InAlGaP "	
197	" InAlAsP "	
198	" ITO "	
199	"Pentacene "	
200	"Alq3 "	
201	"TPD "	
202	"PPV "	
203	"Organic "	
204	"Ba2YCu3O7 "	
205	"Ba2NdCu3O7 "	

Note: The number symbol(#), the equal sign(=), the single quote (') and the space symbol () must be quoted.

9.13.31: GENERIC PARAMETER - PATTERN

On black and white displays, pattern fills are used instead of colors. There are 18 patterns numbered 0 to 17. Pattern 0 (solid) and pattern 17 (empty) are normally not used. Each material has a default pattern, which will be used by the region if no pattern is set.

This page is intentionally left blank.

10.1: Introduction

MASKVIEWS is an IC layout editor designed to interface IC layout with Silvaco's process simulator. It can draw and edit IC layout, store and load complete IC layouts, and import/export layout information using the industry standard GDSII and CIF layout format. MASKVIEWS provides layout information to the simulators, enabling any part of the layout to be simulated. Currently supported simulators are:

- SSUPREM3: A 1D process simulator. MASKVIEWS provides an array of switches specifying whether each mask is present or absent at any selected point on the layout.
- ATHENA: A 2D process simulator. MASKVIEWS provides a set of mask regions for each layout level giving the start and end points of masks on any arbitrary cross section on the layout. MASKVIEWS also provides information on how ATHENA should construct its grid. It also specifies region names for use with DECKBUILD's **Extract** parameter extraction command and specifies names to be used as electrodes when passing the simulated region on for device simulation.
- OPTOLITH: A 2D lithographic extension to ATHENA. MASKVIEWS provides a 2D set of mask rectangles to be simulated by OPTOLITH (rectangles are generated even if they are not drawn on the layout). MASKVIEWS also allows phase and transmittance values to be specified for each mask element.

MASKVIEWS also provides features to allow layout experimentation such as:

- mis-alignment
- polygon oversizing/undersizing
- global rescales
- region definition — depending on combinations of present mask elements.

Productivity enhancements, such as zoom and pan, full on-line help and manual, and user specific start-up properties are also available.

10.2: Starting

MASKVIEWS can be used directly from DECKBUILD as a support tool, as a stand-alone UNIX utility, or from within the VIRTUAL WAFER FAB (VWF) environment.

Note: When starting MaskViews for the first time, follow the procedure titled **Starting For First Time in From DeckBuild** paragraph.

10.2.1: From DeckBuild

You can start MASKVIEWS from DECKBUILD by selecting **Tools→Start Maskviews....** A popup will appear, which lists all of the layout files in the current directory (if any). You can change the directory name and search string, and you can select a layout file for loading from the list. Clicking on the **Start MaskViews** button executes MASKVIEWS and loads the selected layout file. See “Starting MaskViews With An Example Layout File” on page 10-3 to learn how to load the provided example layout files. The **MaskViews Base Window** (Figure 10-1) appears after a short period.

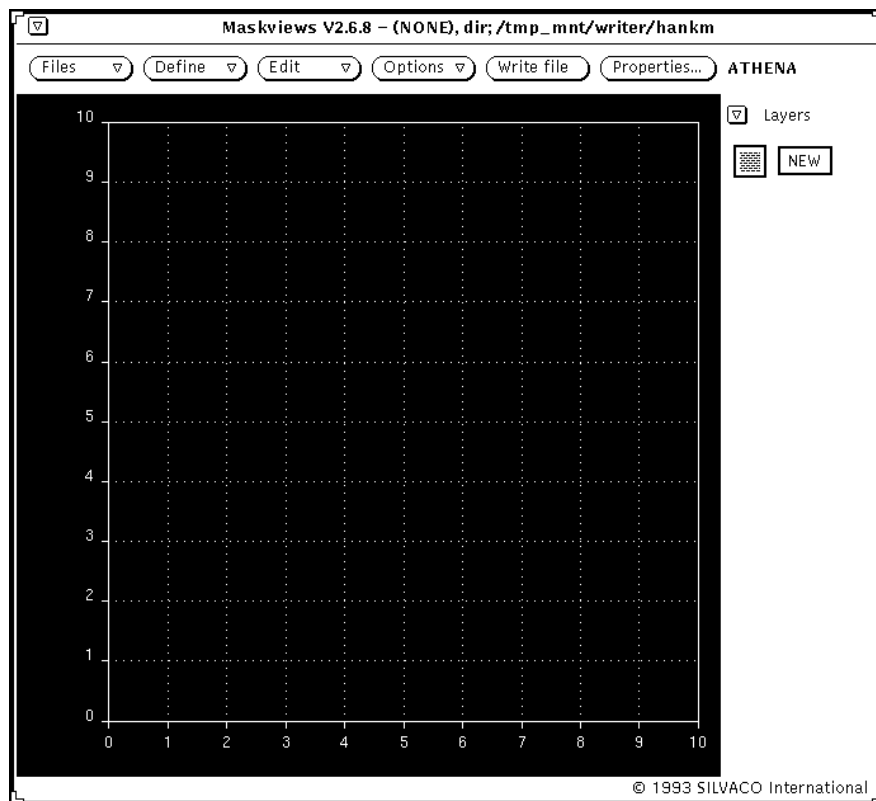


Figure 10-1: MaskViews Base Window

Starting MaskViews With An Example Layout File

If you haven't used MASKVIEWS before, you may find it useful to load and later modify one of the provided example layout files. Examples can be invoked from DECKBUILD's **Main Control** menu. In the following example, a demonstration input deck is loaded into DECKBUILD, and a cross-section layout file is copied to the current working directory.

1. To load the demonstration input deck to DECKBUILD, open the DECKBUILD **Main Control** menu and select **Examples**. When the DeckBuild Examples window appears, select **ATHENA: Examples Including Process, Topography and/or Lithography** and a list of input decks will appear.
2. Double click on **anex18.in: Simple CMOS Example Using MaskViews** and a brief description of the input deck will appear. When the mvanex01.in window appears, click on **Load example** button. Observe the example input deck loaded into DECKBUILD text window.
3. Pull down the **DeckBuild Tools** menu and select **MaskViews→Start MaskViews...**.
4. The file "anex18.in" should be listed in the filebox of the MASKVIEWS Layout Files popup. If not, click on **Refresh** and click on **Start MaskViews**. The MASKVIEWS window will then appear.

Interfacing With A Simulator

To load the layout information into **DeckBuild**:

1. Click on the **Write** file button and define a cutline area by clicking on a start point. Then, click on an end point. This action causes the **MaskViews:ATHENA** cutline popup to appear. Click on **Write** and observe preview cutline popup (2D masks from 2.2,11.6 to 7.8,10.9) appears.
2. Pull down the **DeckBuild Tools** menu and select **MaskViews→Cut files...** to observe the **MaskViews Cut Files** popup. You can then use one of the following methods to load the example cutline file: **Disk Files** or **Drag & Drop**.

To use **Disk Files**:

1. Pull down the **Category** menu.
2. Select **Disk Files**.
3. Select **Load**.

To use **Drag & Drop**:

1. Pull down the **Category** menu and select **Drag & Drop**.
2. Press SELECT on the icon in the upper left corner of the preview cutline popup (2D masks from 2.2,11.6 to 7.8,10.9), then drag it to the **MaskViews Cut Files** popup window and release.
3. Click on the icon to frame it and click on **Load**. The filename `default.sec.xx` has been loaded, where `xx` is an incremental number, depending upon the number of cross section *cutline files* in the current working directory.

To load this cutline file into DECKBUILD:

1. From DeckBuild's main window, open the **Tools** menu.
2. Select **MaskViews**
3. Select **Cut files** and specify a file name.

10.2.2: Using MaskViews Inside The VWF

When used within the VIRTUAL WAFER FAB (VWF) AUTOMATION TOOLS environment, some of the operations of MASKVIEWS are made slightly different. To start MASKVIEWS, double click on one of the layout mask items in the **Masks** directory in the VIRTUAL WAFER FAB library section. Layout entries are created using the **Create** option while displaying the **Masks** directory. A layout entry must be present to start MASKVIEWS, even though it may be empty. You can copy a layout entry by starting MASKVIEWS on the destination layout item, then dragging and dropping the source item to the MASKVIEWS layout window. A warning is displayed if any mask elements already exist and are deleted by the copy action. The **Files** control button is replaced by a **Data** button, whose default action is to store the layout back into the database. All of the normal file control facilities are available on a submenu contained under this button. The **Write File** simulator control button is retitled **Write data**. Pressing this will display a popup from VIRTUAL WAFER FAB asking you to create the name of the mask data entry. The **Write to deck** options are not available when used within the AUTOMATION TOOLS.

10.2.3: Starting MaskViews From The UNIX Prompt

To start MASKVIEWS from the UNIX command line prompt, enter:

```
maskviews (command line options)
```

Command line options are a series of switches that causes the initial MASKVIEWS displayed to be set different from the default. File related options take two parameters and are:

- **-d** (*file name*): sets the output default to the named file.
- **-f** (*file name*): loads the named layout file.

The following option loads the named ATHENA grid template:

- **-g** (*file name*)

Options are available to adjust certain display parameters are as follows:

- **-mono**: force monochrome display mode.
- **-olwm**: format poppies for the OpenLook window manager (normally the default).
- **-mwm**: format popups for the Motif window manager.
- **-usa**: use American spellings throughout (normally default).
- **-uk**: use English spellings throughout (default in GMT time zone).

The default target simulator can also be set from the command line using the switches:

- **-suprem3**
- **-athena**
- **-optolith**

10.3: MaskViews Main Window

10.3.1: Layout and Functionality

The **MaskViews Base Window** (Figure 10-2) display is composed of up to four sections.

- The **Layout** window is the layout area where all layout and simulation actions are performed. Unless MASKVIEWS was started with a layout file, the layout window is an empty grid.
- The **Control** panel is displayed along the top of the screen and contains a series of buttons, menus, and options, which are used to control the actions of MASKVIEWS.
- The **Key** panel is displayed on the right side of the window. The key panel has several modes and its function change as the mode is changed.
- The **Footnote** panel is displayed across the bottom of the window and shows any messages relevant to the current actions.

The filename of the loaded layout file is shown along the top title bar of the window. "NONE" is displayed if no file has been loaded. Whenever you make changes to the layout without saving them, the word "edited" appears in the title bar.

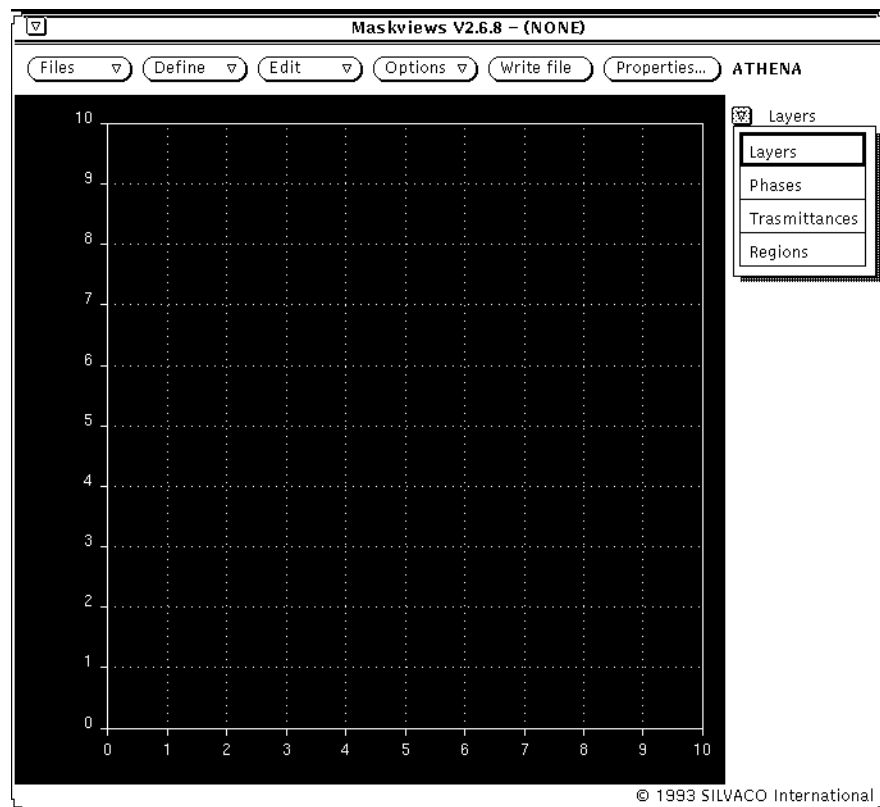


Figure 10-2: The Mode Menu of MaskViews Base Window

Control Panel

Commands are performed by selecting the appropriate option from one of the following menus or buttons displayed on the command panel.

- **Files** contains a menu of file functions, print functions, and the empty (or delete all) function.
- **Define** lists all of the layout, screen, and object definition options.
- **Edit** contains all of the polygon edit commands.
- **Options** contains optional utilities such as zoom and pan, on-line manual, and release notes.
- **Write** file creates the cross-section file(s) that is used with simulators.
- **Properties** displays the user-configurable Properties window.

Key Panel

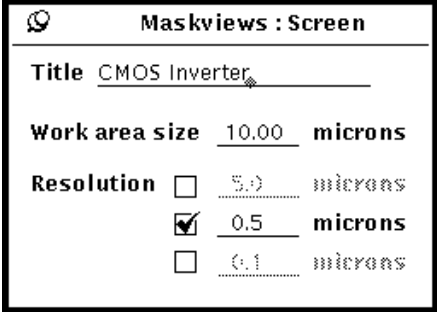
There are four object display modes in MASKVIEWS: **Layers**, **Phases**, **Transmittance**, and **Regions**. These are selected from the menu at the top of the key panel.

- **Layers** — The layout levels in the structure are displayed. The current edit layer is chosen by selecting the layer name from the list. Levels are made visible/invisible by selecting/deselecting the layer key item next to the layer name. Selecting a layer while holding down the SHIFT or CONTROL keys causes all other layers to be disabled or enabled respectively. If the number of layers in the layout becomes too great to be displayed down the side of the screen, then only some of the layer keys are shown. Nudge up and down buttons are then displayed, allowing the layers keys which are to be shown to be scrolled up and down.
- **Phases** — The layout screen displays masks from one level only. The key panel shows phase shift values for each mask on this level. The current default phase shift value used for editing is set by selecting an item in the key, or by adjusting the phase value slider below it.
- **Transmittances** — This mode is similar to the phases mode except the key and the display (Figure 10-5) shows mask transmittance values.
- **Regions** — This shows regions that have been set up in the **Region definition** popup. All polygons are outlined only on the screen but can still be edited. The current edit layer is chosen by selecting the layer name from the menu on the key panel. Regions are enabled/disabled by selecting the **key** button next to the region name. Regions are features of the layout that can be identified using layer combinations. For example, device gate regions are areas where polysilicon overlaps active areas.

10.4: Editing

10.4.1: Defining Edit Parameters

Selecting Screen from the Define menu presents the **Screen** popup (Figure 10-3).



Maskviews : Screen		
Title	CMOS Inverter	
Work area size	10.00	microns
Resolution	<input type="checkbox"/>	5.0 microns
	<input checked="" type="checkbox"/>	0.5 microns
	<input type="checkbox"/>	0.1 microns

Figure 10-3: Screen Popup

This popup is used to set the following screen parameters:

- **Title** — Sets a title string that will be displayed at the top of the layout and on any hard copies.
- **Work Area Size** — Sets the size (in microns) of the total layout area.
- **Resolution** — Specifies the resolution to which all draw, edit and simulator actions are rounded. Three input fields are available to specify the resolution. The check box next to each is used to define what is currently active. This allows easy selection of different resolutions for different types of work.

The values specified in these fields are stored in the layout files. These can be set to different values depending on the simulator selected. They cannot be stored as defaults. See Section 10.8.2: “Default Properties” for a description of the other options that affect layout and editing on the screen.

10.4.2: Defining Layout Layers

IC layout descriptions within MASKVIEWS are specified in terms of polygons that exist on a number of layout layers, each of which corresponds to a reticle mask used in the IC fabrication process. Layers are defined by using the **Layers** popup (Figure 10-4) displayed by selecting **Define→Layers...**.

Maskviews : Layers

Current layer :

Label : NEW Name : undefined

Field Electrodes

Mis-alignments : Delta CD :

x 0.0 y 0.0 0.00

	Thickness	Minimum spacing		Divisions
		Spacing	Distance	
Resist	<u>1.00</u>	<u>0.50</u>	<u>(.)(.)(.)</u>	<u>1</u>
Barrier	<u>0.01</u>	<u>0.005</u>	<u>0.000</u>	<u>1</u>

Figure 10-4: Layers Popup

The values displayed in the popup relate to the attributes of the current edit layer. The current edit layer can be changed by selecting a new layer name on the key panel. The **Layers** fields are the following:

- **Label** — This field provides a short abbreviated (five character) name for the layer, which is used on the key canvas and to identify the layer on any generated outputs. You must press the Return key after you have entered a value for this field.
- **Name** — This field contains a more complete descriptive name for the layer. You must press the Return key after you have entered a value for this field.
- **Field** — This field indicates whether the mask reticle is a dark or clear field. Dark field layers consist of holes cut into an opaque reticle. Clear field layers have opaque mask elements on a transparent reticle. The field value is used when writing mask output to determine if a present mask polygon corresponds to the presence or absence of mask material.
- **Mis-alignment** — These fields (x and y) are used to offset a complete mask layer in the horizontal and vertical directions. This can be used to experiment with the effects of accidental or deliberate mask mis-alignment.
- **Delta CD** — This field value is used to bloat or shrink mask elements by the value specified. Entering a value causes the sides of the each element to move outwards in inwards by the difference between the old value and the new value. If the **field** attribute is **Clear**, then a positive change causes the elements to bloat. A positive change causes the elements to shrink if the field is **Dark**.
- **Add Button** — New layers are added by selecting one of the options available under the **Add** button. The new layers can be inserted before or after the current edit layer.
- **Delete Button** — The current edit layer can be deleted by clicking on the **Delete** button. A warning message is displayed if the layer currently contains polygons.

If the target simulator is set to ATHENA (see Section 10.8: “Properties”), the following additional items are displayed on the popup.

- **Electrodes** — This item is used to indicate that the current layer is a conducting layer and mask label names used will be passed on to the simulator as electrode names.
- **Thickness** — Deposited mask material thickness is specified in the thickness field, and its material type can be set to either **Resist** or **Barrier**. This information is used with the ATHENA `deposit` statement (see the ATHENA USER’S MANUAL) to form the masked layer. If the target simulator is set to SSUPREM3, then the resist thickness can again be specified but its type cannot.

Selecting **Define→Biases...** makes an alternative method of accessing the Delta-CD values available. The bias value listed in this popup is identical to the Delta-CD value listed in the layers popup. The complete set of biases/Delta-CD’s can be loaded from and saved to files using the **File loader** popup displayed by selecting **Files→Biases...**

10.4.3: Drawing Objects

All masks objects used inside MASKVIEWS are polygons, which may be irregular and contain up to several hundred sides. Most drawing and editing is performed in terms of polygons. Facilities, however, are provided to allow you to draw in terms of more regular shapes. The **Objects** popup (Figure 10-5) controls the current shape drawing mode. To display this popup, select **Define→Objects...**. The **Object** field on this popup indicates what type of shapes are to be drawn on the current edit layer. Initially, three options will be available: **polygon**, **stick**, and **serif**.

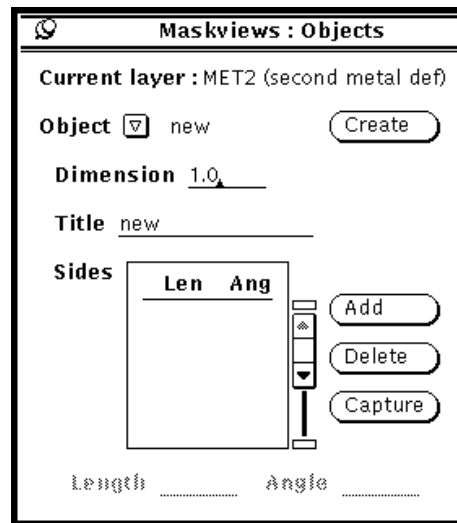


Figure 10-5: Objects Popup

Polygons

When the object type is set to **polygon**, the mask objects are drawn as multi-sided polygons of any shape. A polygon is drawn by selecting the vertices of the polygon on the main layout screen. After the selecting first vertex, a line will be drawn connecting the most recent vertex with the current mouse pointer position.

A polygon is completed either, by closing the polygon path, by selecting the first point on the path again as the last point, or by selecting the **Edit→Close path**. You can cancel polygon drawing any time before the path is completed by selecting **Edit→Cancel**. Polygon drawing obeys the rules for resolution spacing set up in the screen definition popup, and the angle constraint rule set up in the **properties** popup. If you use the **Close path** option to complete a polygon, then the drawn path from the last selected vertex to the first vertex also obeys these rules, which may cause extra vertices to be added.

Sticks

Sticks are used to draw single line polygons. Sticks are constructed by selecting the start and end points of the stick on the main layout screen. A stick is drawn along the line joining the two points with a diameter as specified in the **Width** field of the objects popup and a shape as specified in the **Type** field. Three stick types are available:

- **Butted** — The end points of the stick exactly coincide with the end points of the drawn line. The stick is still rectangular.
- **Extended** — The end points of the stick overshoot the end points of the drawn line by half of the diameter at both ends. The stick is still rectangular.
- **Rounded** — The end points of the stick overshoot the end points of the drawn line by half of the diameter at both ends. The ends of the stick are round.

Sticks are converted to polygons as they are drawn. You cannot alter the width, type and position of the stick once you draw the stick. You can cancel stick drawing before specifying the end point by selecting **Edit**→**Cancel**. Stick drawing obeys the rules for resolution spacing and angle constraint. If you use MASKVIEWS with the OPTOLITH simulator, then a warning message will be displayed if the stick diameter selected cannot be correctly quantized with the specified resolution.

Serifs

Serifs are small squares that are added to masks to sharpen the corners of the mask when it is projected and exposed on the semiconductor substrate. When in this drawing mode, a single click places a square with each side specified by the **width** field centered at the mouse click position. Once serifs are placed, their size cannot be changed. The serif size set here is also used as the size for serifs when added using the **Add serifs** edit menu option (see Section 10.4.5: “Polygon Editing”).

10.4.4: User Defined Objects

User defined objects allow any polygonal shape to be drawn on the current layer by specifying the path in terms of distances and angles. The shape is then drawn by selecting the first **Vertex** point of the shape on the main layout window. User defined objects are created by selecting the **Create** button on the Objects popup. This is then appended an object titled new to the end of the objects list. Once you select the object as the current object type, you can enter a more descriptive name for the object in the **Title** field of the popup. Sides of the shape are described in terms of angles and lengths, which are listed in the scrolling list on the Objects popup. The angle value specifies a rotation to the path direction to be applied before drawing the side. The initial angle is vertically upwards on the screen, and angle values are in degrees clockwise. For example, the length and angle pairs (1,0), (1,90), (1,90), and 1,90 define a unit square which could be used to define contact holes on the layout.

The length of each side is determined by the side length parameter multiplied by the object dimension parameter. Therefore, you can create similar polygons of different sizes by changing the dimension parameter only. The polygon path is always closed even if the first and last points in the list do not coincide. Extra sides can be added after the current side by selecting the **Add** button on the Objects popup. Sides can be deleted by selecting the **Delete** button.

You can encode objects already drawn on the layout screen as user defined objects by clicking on the **Capture** button on the Objects popup and selecting the desired polygon on the layout. The drawn polygon must be on the current edit layer. The object is then encoded so that it will be redrawn again with exactly the same size if the current dimension value is used.

User objects are converted to polygons as they are drawn. Once drawn, their original attributes are lost and cannot be altered. User objects do not obey the rules of resolution or angle constraint. Any shape can be drawn, some of which may not be totally suitable for the simulator being used. User object placement does conform to the resolution spacing. A library of useful shapes (e.g., circles or octagons) has been installed in the `$SILVACO/var/maskviews` directory.

10.4.5: Polygon Editing

The **Edit** menu contains command options used during the edit stages of creating MASKVIEWS layouts. The options are:

- **Cancel** — Aborts the current edit action before any changes are made.
- **Move** — Allows a polygon to be moved. Select and drag the desired polygon to its new position and release the mouse button.
- **Copy** — Allows a polygon to be copied. Select and drag the desired polygon to the new position and release the mouse button.
- **Invert** — Mirrors a polygon on the current edit layer, this can be either horizontally or vertically.
- **Mask** — Allows attributes for individual mask elements to be altered. After selecting, point and click on a mask element on the current edit layer. If the edit mode is phases or transmittances, then the respective values for the element can be altered using the items on the key panel (described in more detail in Section 10.4.6: “Object Editing”).

If the edit mode is **Layers**, then a popup is displayed as shown in Figure 10-6. This allows the element to be moved to a different layer or resized or both by the specified amount. Clicking on the **Apply** button puts the changes into effect.



Figure 10-6: Mask Alter Attributes Popup

- **Label** — Attaches labels to polygons for use as electrode names. A popup is displayed with a text field requesting label name once a mask object has been selected. This popup also contains a checkbox enabling/disabling the whole layer as a layer of electrodes. This option copies the electrode definition from the Layers popup.
- **Close path** — This is used to complete a partially drawn polygon.
- **Group** — These options are used to **cut** and **paste** sections containing many polygons in single actions. After selecting the **cut** or **copy** options, an area of the layout is marked by selecting opposite corners of a “rubber-band” box. Polygons contained either partially or wholly within this box are then copied to the group paste buffer. If the cut option was chosen, these polygons are also removed from the layout screen. After choosing **paste**, the drag operation can be used to position the group paste buffer back onto the layout screen. The **undo** option can be used to undo a **cut** or a **paste** operation, as long as no other actions have been performed since the operation.
- **Slice** — Cuts all polygons along a line into sections. After selecting, a line is drawn on the screen and after the second point has been defined all polygons along the line is sliced. This can be used to assign more than one electrode name to a given polygon. If you cut a polygon, its labels are then discarded.

10.4.6: Object Editing

- **Merge** — This is used to merge two overlapping polygons into one shape. After selecting **Merge**, two polygons are chosen for merging. A warning message is displayed if the polygons cannot be merged correctly. The merge operation uses only the outside surfaces of the merged polygons, any totally enclosed spaces are lost.
- **Mode** — These options are used to select whether all drawing actions create new polygons or cut holes in current objects. If you select **draw outline**, then all drawing operations create new mask objects. If you select **cut hole**, then drawing operations are used to cut holes in existing objects. In order for hole cutting to succeed, all vertices and all sides of the cut polygon must line within the existing mask object.
- **Electrode** — For certain simulator modes, this allows electrodes to be specified by defining two end points. After selecting the option, select two point for the electrode. If the two points differ in by X and Y coordinate, a rectangular electrode is generated. Otherwise, a straight line electrode is formed. In both cases, the electrode end points move to coincide with vertex points of the closes polygon.
- **Auto label** — Selecting this option attempts to automatically label polygons with labels loaded from a GDSII file. Polygons are automatically labelled if a label position lies within the polygon and the polygon does not already have a label.
- **Add serifs** — Adds small square serifs at all vertex points of a selected polygon. After selecting this option, select a polygon for the serifs to be added to.

Note: Serifs can not be added to serifs.

- **Undo** — When enabled from the **Properties** section (see Section 10.8: “Properties”), this allows you to cancel the most previous `edit` command. Once you perform an **undo**, the option becomes unavailable until a further edit action occurs.
- **Delete** — Erases a mask object.

10.5: Simulator Control

10.5.1: Overview

MASKVIEWS' main function is to provide process simulators with layout data, enabling any point on a real layout to be simulated without having to manually calculate mask edges. MASKVIEWS can either write a summary file for later use (either by DECKBUILD or the simulator directly), or can modify an existing simulator input deck with the statements required to perform the masking actions. To open the popup that controls the output destination, select **Files→Output...**. The popup contains a list of all files in the current directory matching the **Filter** field. The **Write to deck** option is used to select the type of output to be created. If set, then the file named in the **File** field must be a valid simulator input deck. The output is then a modified version of this deck containing the mask information. If **write to deck** is not set, then the output file written only contains the mask information and no input file will be required. The output file name has a unique integer number appended to prevent overwriting existing files.

10.5.2: SSUPREM3

When the target simulator is set to SSUPREM3 (see Chapter 4: “DeckBuild”, Section 4.3: “Invoking DeckBuild”), MASKVIEWS generates a 1D set of mask elements corresponding to a single point on the layout. After clicking on the **Write file** button, a single point on the layout is selected and the output will be written.

The input deck should be a valid SSUPREM3 deck with the line

```
mask = "abbreviated _layer_name"
```

added where mask information is required. The abbreviated layer name should be one of the names shown on MASKVIEWS key panel. If a mask polygon is present on the layer at the point, and the layer field attribute is set as **Clear**, then the line:

```
deposit photoresist thickness={value}
```

is inserted in the deck at this point. The photoresist thickness value inserted is the value specified in the layers **definition** popup. The deposit statement is also inserted if no masks are present and the field is **Dark**. A strip statement can be used later in the deck to remove the photoresist. If the statement:

```
mask = "abbreviated_layer_name" reverse
```

is found in the deck, then the effect of the field attribute is reversed. A **Cutline/section** for use with DECKBUILD is written after a point on the layout has been selected. A summary mask display is shown if the **Display masks** property has been set. An information message is displayed indicating the name of the output file written.

10.5.3: ATHENA

If you set ATHENA as the target simulator, MASKVIEWS generates a series of etch regions along a line corresponding to the presence and absence of mask elements at those points on the line. After clicking on the **Write file** button, the two end points of the simulated section are selected on the layout.

If you enable the **Cutline edit** property, a popup will appear showing the coordinates of the selected line and three buttons labeled **View**, **Write**, and **Done**. The selected line remains on the layout and can be moved by selecting new end points on the layout. The nearest current end point to the new selected point will be moved. The **View** button is used to draw a summary display of the masks along the line. **Done** cancels the write file operation. Click on **Write** to write output. If the **Cutline edit** property is not enabled, the output is written once the second end point is selected.

The **View** button has two options (**Full Grid** and **Edges Only**) that allow you to modify the grid display. If you select **Full Grid**, a summary display will appear with the initial grid lines that are generated by ATHENA. If you select **Edges Only**, grid lines that coincide with mask edges are displayed.

Note: You should enable grid display and specify an appropriate display mask value in the Properties window before attempting to modify the grid display using the **View** button options.

If you set the **Request Filename** property, you must specify the name of the output file. If **Cutline edit** is enabled, then an extra field labelled **Filename** is available to specify the name. If **Cutline edit** is disabled, then a popup is displayed requesting the output file name after the second cutline point has been selected. You can alter the values of the coordinates on the end points of the cutline by changing the values shown on the **Cutline edit** popup.

The input deck should be a valid ATHENA deck with the lines

```
mask = "abbreviated_layer_name"
```

added where mask information is required. The abbreviated layer name should be one of the names shown on MASKVIEWS **key** panel. At runtime, DECKBUILD replaces this line with:

```
deposit photoresist thickness={value}
```

or with the line:

```
deposit barrier thickness={value}
```

The material type and thickness written are those defined in the Layers popup. The deposit statement is then followed by a series of etch statements. A region to be etched is any area not containing a polygon mask on a clear field layer or any area containing a mask on a dark field layer. The etch statements will appear similar to:

```
etch photoresist start x = {start of region} y = -20.0
etch continue x = {end of region} y = -20.0
etch continue x = {end of region} y = 20.0
etch done x = {start of region} y = 20.0.
```

If the statement:

```
mask = abbreviated layer name reverse
```

is found in the deck, then the effect of the field attribute is reversed. If you disable **write to deck**, an information file for use with DECKBUILD will be written after selecting both points on the layout. A summary mask display will appear if you set the **Display masks** property. You can load the cutline directly into DECKBUILD by dragging the icon displayed in the top corner of the summary. See Section 10.8.5: "Drag and Drop" for more information. An information message will appear indicating the name of the output file written.

Grid Definition

MASKVIEWS also controls the initial grid generation statements for ATHENA. Definition of the grid control parameters is done using the grid control popup (Figures 10-7 and 10-8) displayed by selecting **Define→Grid→x...** and **y...**. The **Horizontal grid** popup shows the grid definition parameters for the current edit layer, and a list of all layers in the layout. Layers must be enabled in the list if they are to have any effect on the grid.

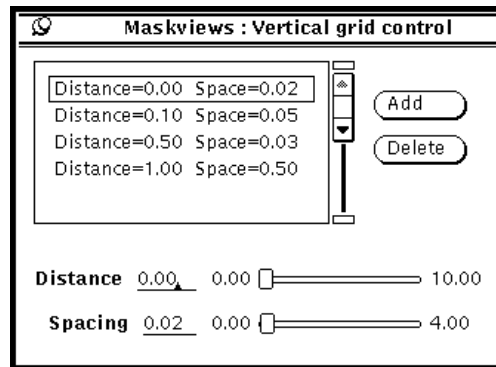


Figure 10-7: Vertical Grid Control Popup

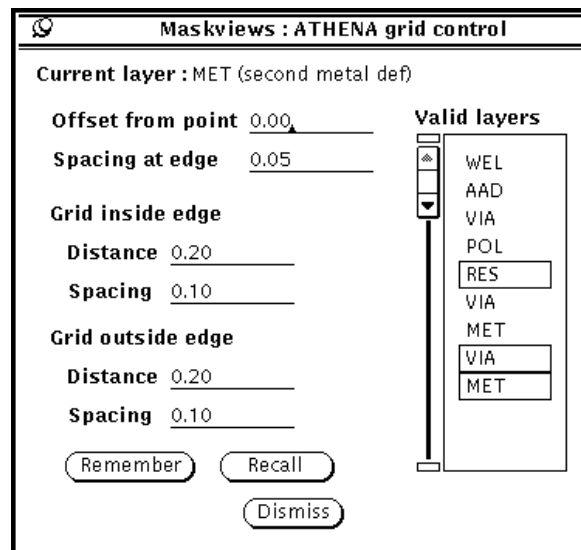


Figure 10-8: Horizontal Grid Control Popup

The **Remember** and **Recall** buttons on the **X grid definition** popup allows the grid spacing parameters for single layers to be remembered and recalled for later use. Clicking on the **Remember** button stores the currently displayed settings of spacing parameters to memory. Clicking on **Recall** restores values to the popup from memory. These can be used when temporarily changing values to observe the effect of spacing or to copy spacing values from one layer to another.

Grid definition in ATHENA consists of a series of statements:

```
line x loc={location} spac={spacing}
```

which defines grid spacing at a specific location. MASKVIEWS generates line statements for each mask edge on enabled layers. ATHENA requires a grid line to be present at an etch point. It can also specify spacing points inside and outside of the mask region (Figure 10-9). This allows fine grids to be created where known mask edges lie decaying to courser grids at some point further away.

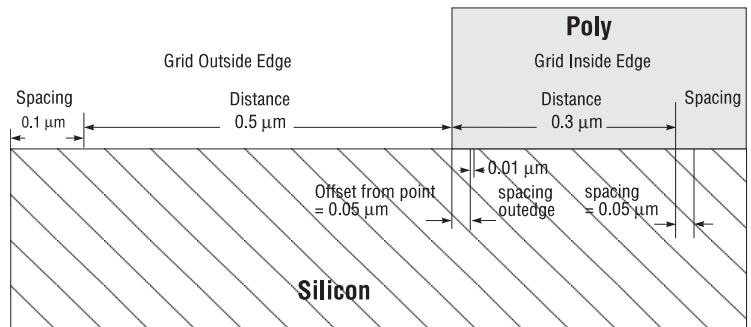


Figure 10-9: Horizontal Grid Spacing Parameters

The **Vertical grid** popup allows vertical distance and spacing lines to be modified using the **Distance** and **Spacing** slides. A line is added to the list by clicking on the **Add** button, and the currently selected item is removed by clicking on **Delete**. The list is automatically sorted by increasing distance. If you enable the **cutline edit** mode, the Cutline popup will contain a maximum depth field, which you can use to specify the largest line y distance. If **write to deck** is not enabled, grid information is written to the output information file in a form readable by DECKBUILD.

Layout Experiments

Layout experiments can be performed with ATHENA decks using the **Loop variables** popup (Figure 10-10) displayed by selecting **Define**→**Looping...**. This allows mis-alignments and delta cd values to be ramped across ranges to create matrices of input decks.

To enable a variable, use the check-boxes along the left side of the popup. You can then enter the layer name, parameter, start, step, and number of steps values for that variable using the displayed fields. A warning message will appear if you use a layer name/parameter combination as more than one variable. The write file sequence is the same as when looping is not active, except many files are generated. If you enable the **Mask Display** property option, masks sets will appear for all the loop points.

Maskviews : Loop variables

	Layer	Parameter	Start	Step	Steps	
<input checked="" type="checkbox"/>	<input type="checkbox"/> POLY	<input type="checkbox"/> Mis-align x	0.0	0.0	1	
<input checked="" type="checkbox"/>	<input type="checkbox"/> MET1	<input type="checkbox"/> Mis-align y	0.0	0.0	1	
<input type="checkbox"/>	<input type="checkbox"/> WELL	<input type="checkbox"/> Mis-align x	(.)	(.)	1	
<input type="checkbox"/>	<input type="checkbox"/> WELL	<input type="checkbox"/> Mis-align y	(.)	(.)	1	
<input type="checkbox"/>	<input type="checkbox"/> WELL	<input type="checkbox"/> Mis-align x	(.)	(.)	1	

2 variables, 4 passes, 2 active layers

Figure 10-10: Loop Variables Popup

10.5.4: OPTOLITH

If OPTOLITH is set as the target simulator, the output consists of a series of 2D mask elements taken from one of the layout layers. The OPTOLITH module of ATHENA works with rectangular elements only and any layout is converted to rectangles before the output is written. Therefore, it is advised that the 90° only angle constraint property is used when targeting MASKVIEWS for OPTOLITH (sloped lines will be converted to a series of steps). As the OPTOLITH module simulates only masks from one layout layer and is interested in phase and transmittance values for each mask element, it is most useful to use either the **Phase** or **Transmittance** edit modes when OPTOLITH is the target simulator.

When you select **Write file**, a popup will appear that shows the coordinates of opposite corners of the simulation area (initially both will be blank) and the name of the output file to be written. The corners of the simulation area are selected by pointing to the location on the layout display and pressing the SELECT mouse button. After selecting the first corner, a “rubber band” box will be drawn from its location to the current mouse pointer location showing the region to be simulated. Once you define the second corner, two more buttons will appear on the popup.

- **Write** — Writes the mask element rectangles to the file specified in the **Filename** text field. Enter a new name in this field if the desired output file name is different than the one indicated.
- **Abort** — Cancels the write file mode and dismisses the popup.

Phase and Transmittance Values

The OPTOLITH photo-lithographic simulator has the ability to simulate the effect of mask elements with different phase and transmittance values. MASKVIEWS provides the facility to edit these attributes on mask elements using the special modes selected on the key panel. When you select one of the two modes, mask polygons from one layout level only will appear with their phase or transmittance values indicated by colors listed in the key. To edit the attributes of a mask element, first select **Edit→Mask**, then point and click to the mask element that needs to be edited. All other elements on the screen are then re-drawn in a dimmer color and the key buttons and sliders are updated with the mask element's phase and transmittance attribute values. You can alter the values for the selected mask by moving the sliders or selecting a new value from the key buttons. Once you set the correct value, you can resume normal editing mode by selecting **Edit→Mask**, then clicking on an empty space on the layout area.

Phase Display Mode

The layout screen displays masks from one level only. The key panel shows phase shift values for each mask on this level. The current default phase shift value used for editing is set by selecting an item in the key or by adjusting the phase value slider below it.

Transmittance Display Mode

This mode is similar to the phases mode except the key and the display (Figure 10-5) shows mask transmittance values.

Serifs

If serifs have been applied to a polygon, they alter the mask image for that polygon when it is written to the file. Serifs added to convex vertices (i.e., outside corners) add their area to the shape of the polygon, while serifs added to concave (i.e., inside corners) vertices subtract their area.

10.6: Files

10.6.1: Loading and Saving Files

All of the data stored in MASKVIEWS can be saved or loaded. File loading and saving is provided through use of a standard File Loader popup (Figure 10-11).

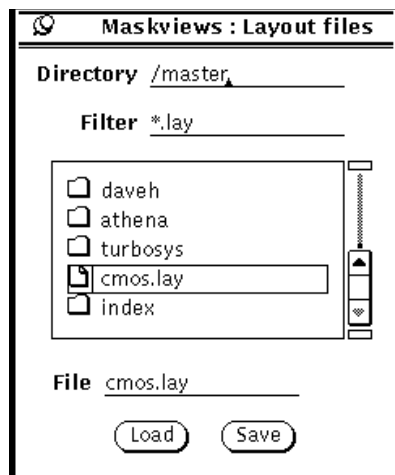


Figure 10-11: File Loader Popup

This popup contains text fields showing the current directory path, a file name filter, and a list of all files that match the filter in the current directory. The list may also show all the directory entries in the current directory depending on the properties setups. The icon type next to the name on the list shows what type of each named item.

Filing system access is provided through options on the **Files** menu (see Table 10-1).

Table 10-1: File Menu Options		
Option	Filter	Files
Layout...	*.lay	layout data
Grid...	*.grid	SSUPREM4 grid templates
Objects...	*.user	user defined polygons
Biases	*.bias	mask biases

Clicking on the **load** or **save** buttons on the popup causes relevant data to be loaded from or saved to file named in the **File** text field. This field is updated whenever you select a file item from the scrolling list or entered manually. Double clicking on a file name in the list also causes the file to be loaded. The current directory can be changed either by typing a new name into the **Directory** field or by double clicking on a directory name in the list. The parent directory is displayed as an upwards arrow labelled **go up a level**. Warning messages appear if you perform a load operation when MASKVIEWS already contains information or perform a save operation to a file that already exists.

10.6.2: Viewing Outline Files

Selecting **Files→Cutlines...** displays a popup that allows you to view previously saved ATHENA mode outline files. Selecting a file in the scrolling list on the popup and clicking on the **View** button causes a line to be shown on the layout where the cross section was taken. If the **Cutline preview** property has been set to **show masks also**, then a summary display of the masks contained in the file are also shown.

To view the outline, it must be saved from the same layout name. If the names are not identical, the outline view is not performed. No changes are taken into account that have been made to the layout since the outline was saved. Editing on the layout screen is disabled, while the outline view popup is being displayed. Dismissing the popup (use the **Done** button) re-enables edit mode again.

10.6.3: GDSII & CIF Import/Export

The GDSII stream (version 6.0) format is an industry standard file format for storing layout information in a file system. It provides a hierarchical system of libraries and layout structures which can contain embedded references to other layout structures. Elements in GDSII format are in terms of polygons or attributed sticks. MASKVIEWS can load individual or hierarchical structures from a GDSII file, converting all data to its internal polygon format. It can also save the current layout as a GDSII library cell and create new libraries within an existing GDSII file.

If you select **Files→import→GDS2 stream format...**, the GDSII stream format interface popup (import/export) will appear (Figure 10-12). This popup has three levels of operation: **Files**, **Libraries**, and **Structures**.

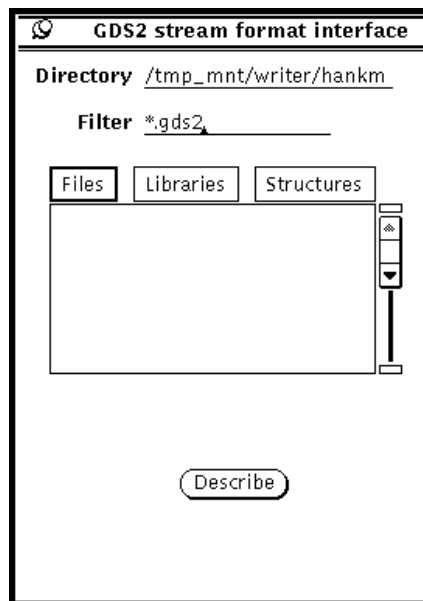


Figure 10-12: GDSII Stream Format Interface Popup

Under the **Files** mode, the list displays all files matching the **Filter** field in the directory defined in the **Directory** text field. Clicking on the **Describe** button with a GDSII file selected generates a popup display describing information held in the file, such as the last access dates and the number of elements contained.

In the **Libraries** mode, all layout libraries in the selected file are displayed on the list. The current library is selected by choosing an item on this list. To generate new libraries, click on the **Create** button and enter a new name for the library in the **Name** field for the newly created item.

The **Structures** mode lists all of the layout structures in the currently selected library. Structures are loaded by selecting an item on the list and clicking on the **Load** button. The **Uncover sub-structures** option is used to specify whether MASKVIEWS should recursively load structures referenced within the selected structure (loading these may take a long time).

The **Offset x and y** fields allow loaded structures to be offset by the specified amounts. As elements are loaded, if they have no stored phase or transmittance values, they are assigned the values currently set as the default phase and transmitter on the key panel. To set the offset, check the box. The default is no offset.

The current MASKVIEWS layout is saved to the library by clicking on the **Save** button. The newly created structure is named in the **Name** field, which may or may not be a currently existing structure name. GDSII files are often very large and operations on them are extremely slow. In an attempt to speed up access, MASKVIEWS loads the complete file into memory and perform all operations on the copy held in memory. Sometimes, it is impossible to load the complete file. In which case, the warning message:

```
File cache failed - this may take some time
```

is displayed and operations are performed directly to the named file, which is much slower.

The minimum feature size that is resolvable with MASKVIEWS should not be less than 1/10,000 times smaller than the size of the whole GDSII structure. That is, structures imported into MASKVIEWS should typically be less than one millimeter along the longest edge.

Technology files are used to add names and other pertinent information to layout layers imported from GDSII files (in which layers are represented simply by their number). Selecting **Files→Import Technology files...** causes a File Loader popup to be displayed, which allows the name of a technology file to be specified. The file specified here is then be used in conjunction with any GDSII read/write operation. The format of the files are lines of the format:

```
layer_number "layer_abbreviation" "layer_description"
```

When the GDSII structure is loaded, if any elements are found to exist on layer `layer_number`, then a layer with the corresponding name and description will be formed in MASKVIEWS. When a GDSII structure is saved, any elements existing of layers labelled `layer_abbreviation`, is assigned to the layer `layer_number`.

CIF is another hierarchical file format for describing layout. You can import/export CIF files by selecting the **Files→Import/Export-CIF format...**, which cause a File Loader popup to appear. The name of the CIF file can be selected here and loaded by clicking on the **Load** button.

If the **CIF file preview** property has been set to **yes**, then a popup is displayed showing the first level hierarchy of the CIF file with numbered boxes indicating the relative position of each structure. To descend into a structure, point to it with the mouse pointer and select the **Zoom in** menu option from the mouse menu. The sub-structures contained within the structure are then displayed. If no substructures exist, then the display is blank. The path leading up to the current structure is displayed along the top of the popup. The hierarchy can be ascended by selecting the **Zoom out** option from the mouse menu. Once the desired structure in the hierarchy has been reached, clicking on the **Load** button loads the data into MASKVIEWS.

10.6.4: Creating GDSII Stream Format

MASKVIEWS can create simple GDSII stream format. If you select **Files→Save GDS2 Format**, the GDSII stream format interface will appear (Figure 10-13). The popup displays all files matching the Filter ".gds" in the directory defined in the directory text field. Choose a file name and click on **Save** and a GDSII stream format file will be saved.



Figure 10-13: GDSII Stream Format Interface

You can also load an existing GDSII file by selecting **Import**. You can then do some necessary editing such as adding labels and saving it to a new GDSII file by selecting **Save GDS2 format**. You can see these electrode labels from MASKVIEWS if you set the simulator to be CLEVER from the **Properties...** menu.

10.6.5: Viewing OPTOLITH Image File

Selecting **Files/import→OPTOLITH format...** opens a File Loader popup that allows you to view an OPTOLITH *.sec file. When you select a valid file, clicking on the **Load** button superimposes the mask image contained in the OPTOLITH file over the top of the current layout. This option can be used to verify the mask image data being sent to OPTOLITH. It can also study the mask image produced by OPTLOTH as a result of an optical correction simulation.

Clicking on the **Dismiss** button on the import popup dismisses the popup and removes the imported image from the layout area.

10.7: Utilities

10.7.1: Regions

Regions are used by DECKBUILD to define areas on IC layout where parameter extraction is to be performed. For example, if a cutline intersects the p-channel gate region, then the command

```
extract name=pxj Xj region = pgate
```

extracts the junction depth under the p-channel gate region.

Regions is a display mode used to view sections of the layout that correspond to boolean combinations of masks that are present or absent. An example of such for a CMOS layout would be an n-gate that could be defined as any layout area where WELL is absent and doping, oxide, and polysilicon are all present with no other masks having any effect. To enter **Region** mode, select **Regions** on the menu at the top of the key panel. All polygons are then drawn as outlines only with the color fill used to indicate any defined regions. The region key is displayed on the key panel. Regions are defined using the popup displayed by selecting **Define**→**Regions...**

The **Region** menu at the top of the popup (Figure 10-14) is used to select one of the available regions for definition. The region is then activated/deactivated using the **on/off** selector and named in the **Title** field. The region mask sets are then defined in terms of **true/false/don't care** selectors displayed next to each layer name. Select **true** when a mask on the layer must be present. Select **false** when a mask must not be present. Select **don't care** to indicate the layout layer has no effect on the region. The **Refresh** button is used to refresh the display after a region has been defined. You can still perform polygon editing when in Region mode. All editing, however, is still performed in terms of layout layers and not as regions.

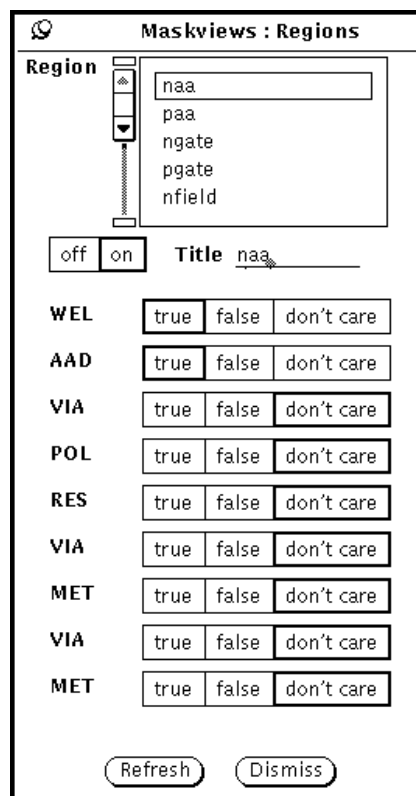


Figure 10-14: Regions Editor

The **Create** button at the foot of the popup creates a new layer. This layer contains the objects formed by combining the other layers in the manner specified in the Regions popup. The polygons on the new layer appear exactly as the shapes shown in the **Regions** view for the currently selected region. The new layer is named the same as the current region. An error is generated if that name is already in use. Objects created by intersecting named electrodes retain the name of any source electrode. If the object is formed by intersection of more than one named electrode, the name given to the created object is one of the names of the source objects.

Note: You must at least specify one layer as 'TRUE' in the Regions popup to create a new layer.

10.7.2: Rescaling

Rescale is used to globally expand to shrink the whole IC layout to study the effect of larger or smaller circuits. Select **Options**→**Rescale layout...** to display **Rescale** popup. This popup has a field containing the rescale value and a **Go** button. Rescale values of 1.0 cause no alteration in the layout. Values less than 1.0 cause the layout to shrink, values greater than 1.0 expand the layout. The circuit size can be returned to its original dimensions by another rescale with the **Rescale** value set to the reciprocal of the first.

The rescaling popup can also horizontally or vertically or both invert the whole layout. The options listed in the **Invert** field specifies which direction inversions are to be applied.

10.7.3: Zoom and Pan

Zoom and pan features are available by selecting **Options**→**Zoom & pan...**. A popup is displayed (Figure 10-15) that shows the full IC layout with zoom factors across the bottom and pan **Nudge** buttons at the side.

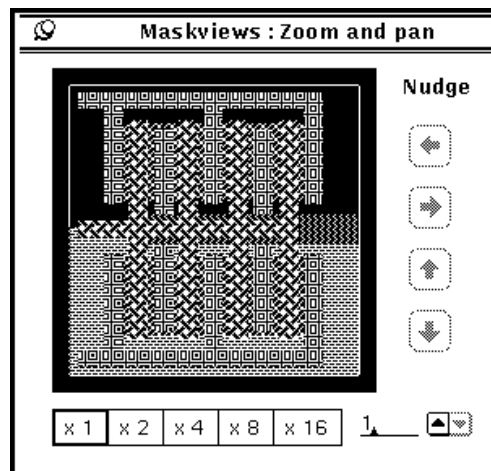


Figure 10-15: Zoom and Pan

Zoom is performed by selecting one of the zoom factors listed or specifying an amount in the numeric field. **X1** causes the whole layout to be displayed. The other options cause only part of the layout to be displayed but magnified. The layout region currently shown on the main display is outlined by a **View** box on the Zoom popup. You can move this view box either by clicking one of the **Nudge** directional arrows to pan up, down, left, or right by approximately 5%, or by selecting a new lower left corner for the **View** box on the zoom display using the mouse pointer.

10.7.4: Reordering Layers

To reorder layers on a layout using the popup displayed (Figure 10-16), select **Options→Order layers....** This allows layers to be re-ordered, copied and deleted.

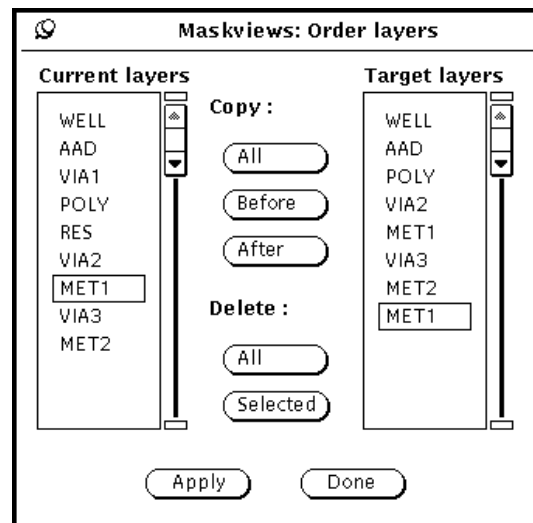


Figure 10-16: Order Layers Popup

This popup has two scrolling lists.

- **Current layers** lists all layers as they currently are in the layout.
- **Target layers** lists the layers in the order they are intended to be. Layers are copied from the **Current** list to the **Target** list using the buttons listed under the **Copy** label.
- **All** copies all of the layers in the **current** list to the **target** list. Any layers previously in the **Target** list are removed.
- **Before** copies the layer name selected in the **Current** list to the position before the layer selected in the **Target** list.
- **After** copies layer name selected in the **Current** list to the position after layer selected in the **Target** list. Layers are deleted from the **target** list using the buttons listed under the **Delete** label.
- **All** deletes all items from the **Target** list.
- **Selected** deletes the currently selected layer in the **Target** list.

Once the layers in the target list have been defined as required, the new layer ordering is put into effect by clicking on the **Apply** button. Elements on layers that are not copied to the target list are deleted. A warning is then issued in such cases giving the chance to abort the reorder. If a layer appears more than once on the target list, then the elements on the layer are duplicated for all layers after the first occurrence. A message is issued if a layer appears more than once warning of such and giving the chance to abort the reorder.

Editing on the layout screen and some of the popups are disabled while the order layout popup is being displayed. The popup can be dismissed without making any changes by clicking on the **Done** button.

10.7.5: Printouts

Hard copy printouts of the complete layout, zoomed region, and the most recent simulation mask set, can all be produced by selecting **Files→Print**. The output consists of a large image of the current layout. A smaller image of the complete layout (if zoom is in operation) with arrows indicating what section of the whole the zoom region corresponds to and the most recent masks summary set if one has been produced. The layout images drawn are normally produced with the same display mode that is currently selected, i.e., **Layers**, **Phases**, or **Transmittances**. Region images, however, cannot be drawn and selecting **Print** while displaying regions have the same effect as printing layers. The destination, file name, and printer type of the hard copy produced are as defined in Section 10.8.6: “Printer Properties”.

10.7.6: On-Line Help

There are two types of help available in MASKVIEW: spot help and the on-line manual. Spot help is available for all GUI items displayed, such as push buttons, sliders and check-boxes. Clicking on the Help key on the workstation keyboard while pointing to GUI item with the mouse pointer opens a popup giving a brief description of the function of the item and how to use it. On keyboards without a Help key, one can be defined using the command:

```
xmodmap -e 'keysym F9 = Help'
```

which would define the F9 function key to be the help button.

The on-line manual is displayed by selecting **Options→Manual...** The on-line manual, which is organized in a two level hierarchical structure, provides an in-depth description of the functions and operation of MASKVIEW. When first displayed, the manual shows an index of all the major section in the help text. The buttons along the top of the help display are used to navigate between the pages of the manual. In other words, selecting **Return to index** causes the initial main section index to be displayed. The **Section** button contains a menu of all the major sections in the manual. If you choose one of these, then the topics for that section will appear. The **Sub-section** button menu can then be used to select a manual page to be viewed.

You print out manual pages by pressing the **Print** button on the help display.

10.7.7: Release Documentation

Selecting **Options→Release notes...** displays a popup listing all of the changes made to the MASKVIEWS manual and program since the previous release. The functionality of this notes popup is the same as for the on-line manual.

Selecting **Options→About MaskViews...** displays a popup that shows the version number and release date of the installed program.

10.8: Properties

10.8.1: Overview

MASKVIEWS has many properties that you can customize. All of these can then be saved to a file and are recalled every time MASKVIEWS is started. To open the Properties popup, select the **Properties...** button on the command panel or click on the **Props** key on the workstation keyboard if one exists. There are several pages of property options in this popup. To see these pages, use the **Category** menu option. Properties are stored by selecting the **Save** button, saved data is stored in the file `$(HOME)/.masterrc`, which is shared between all the VWF INTERACTIVE TOOLS.

10.8.2: Default Properties

The **Defaults properties** section (Figure 10-17) allows you to change the default editing, operations, and setup properties used by MASKVIEWS.

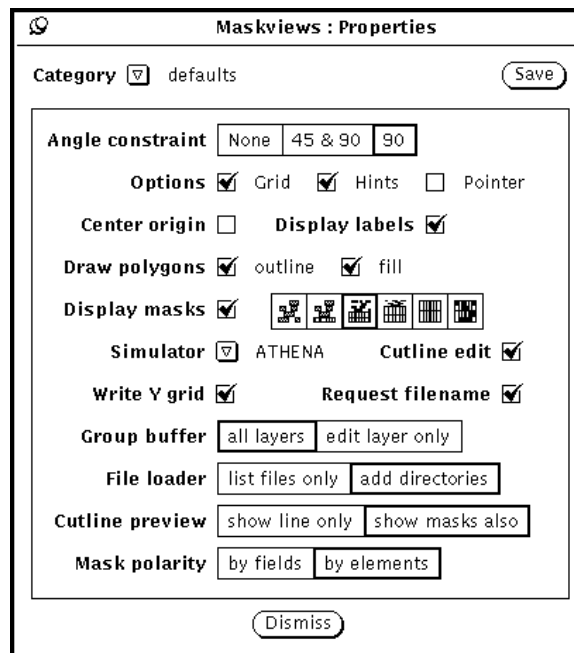


Figure 10-17: Defaults Category Popup

The following choices are available:

- **Angle constraint** is used to limit how polygons sides and lines can be drawn. **None** allows any lines to be drawn. **45 & 90** allows only vertical, horizontal and diagonal lines to be drawn. **90** allows only vertical and horizontal lines.
- **Options** — **Grid** switches on and off drawing of the layout space dashed grid lines. **Hints** enables and disables hint messages which may be displayed at the foot of the main screen. **Pointer** enables and disables the continuous display of the mouse pointer position at the foot of the key canvas.
- **Center origin** specifies whether the 0,0 point lies at the bottom left of the layout screen or in the center. This option may be automatically set by some of the simulator options.
- **Display labels** indicates whether the mask element labels are also displayed on the layout.
- **Display masks** enables or disables the summary display of the mask sets and grid preview generated when writing a simulator output file. If ATHENA is the target, the icon display choice allows you to alter the ratio of grid and mask display.
- **Simulator** defines the target simulator for MASKVIEWS.
- **Cutline edit** selects whether the ATHENA cutline can be moved before writing the output file.

- **Write y grid** enables/disables writing the ATHENA vertical grid information to the output file.
- **Group buffer** specifies whether the group cut and paste operations are performed on all edit layers at once or only on the current edit layer.
- **File loader** sets the option on the file loader popups to display only files matching the filter masks or also list all of the directories contained in the current directory.
- **Request filename** specifies whether a popup will appear for each cutline that requires a filename.
- **Cutline/Grid preview** sets cutline display options. When you load a cutline file for viewing into MASKVIEWS and you set this option to **show line only**, then only a line on the layout will appear. If you set this option to **show masks also**, then the view also shows a mask summary display.
- **Mask polarity** specifies whether the dark/clear field of the **Layers** popup refers to the opaque/clearness of the mask elements or the mask field. Changes made to mask polarity do not take effect until MASKVIEWS is restarted.
- **Undo last action** enables or disables the **undo** command on the main Edit menu. **Undo** is achieved by saving a temporary file between each edit action, which may not be desirable in mask editing if saving the whole layout requires too much time.

10.8.3: Display Properties

The display properties alter the physical layout of the main **MaskViews** screen.

- **Key panel** specifies whether the key panel is displayed on the left or the right side of the layout window.
- **Control panel** selects whether the control panel containing the function buttons and menus is displayed.
- **Screen menu** — When set to **basic**, pressing the mouse MENU button on the layout screen displays the **Edit** menu only. When set to **extended**, the menu displayed contains all of the options normally available on the control panel. When the control panel is switched off, **Screen menu** set to **extended** is enforced.
- **Edit cursor** specifies whether the window edit cursor is the standard pointer type or a transparent cross-hair type.

10.8.4: 3D Mask Properties

This category is used to customize the three dimensional mask summaries which are displayed when 3D Process simulator output is generated by the ATLAS simulator.

- **3D projection** indicates the type of draftsman's drawing functions used to convert the 3D models to 2D images.
- **Draw outlines** specifies whether outlines of all the masks displayed are visible through obscuring mask objects.
- **Block height** specifies that mask blocks drawn should completely extend vertically the space allocated to the layer, or only half of it.
- **Cast shadows** toggles the drawing functions that cause shadows from higher mask objects to be cast onto lower ones. Shadow drawing make take some time to calculate, so is not always desirable.

10.8.5: Drag and Drop

This category defines parameters used with the drag and drop abilities available in MASKVIEWS.

- **Layout drop site** selects whether to drop layout information from the VWF onto MASKVIEW's main layout screen for loading.
- **Cutline drag site** specifies whether to drag cutline files from the summary displays to DECKBUILD.
- **Drag threshold** specifies the number of pixels the mouse pointer has to move with a button clicked before it registers as a drag.
- **Draw drag source** selects whether to draw a drag icon on each cross-section summary display.

- **Invalid drop site** displays the cursor image when the mouse pointer passes over a region where dragged data cannot be dropped.

10.8.6: Printer Properties

The printer properties category is used to select the destination and type of printer to generate the outputs.

- **Destination** selects whether output will be written to a file or sent directly to a printer. When you select **Printer**, a menu will appear listing all of the known printer queues. Select the destination queue in this menu. When you select **File**, a **File** field will appear requesting the name of the file to create. If you need to append the file name with a unique number to prevent an overwrite, use the **Indexed** field.
- **Type** specifies the destination type of printer.
- **Color** check-box will become available if color options are supported for the selected printer type. This option is not available for all printers.
- **Layout** indicates the paper orientation.
- **Page size** selects the printer paper size.

10.8.7: Import Properties

The import properties are used when importing and exporting GDSII structures to encode some of the attributes available in MASKVIEWS, which are not specifically catered for in the GDSII stream format description.

- **Datatype value** is the numeric attribute that will be written as the datatype of masks elements when exporting. **Datatype** is ignored on import.
- **Name attribute** specifies the GDSII PROPATTR tag number that is used to hold the mask labels.
- **Phase attribute** and **Trans. attribute** specify the GDSII PROPATTR tag numbers that are used to hold the mask phase and transmittance values.

You can disable all three attribute options for import/export using the check-boxes displayed alongside.

There are two properties that alter the way CIF format layouts are loaded: **Limit calls** and **File preview**. **Limit calls** defines how many substructure levels are descended when loading a CIF structure. A check box enables/disables the limit, and slider allows you to define the limit. **File preview** selects whether to display the substructure previewer when you press the **CIF file Load** button.

10.8.8: Color Properties

This property section allows the colors used for each edit layer to be changed from their default values. The slider labelled **Number** is used to select a layer number whose colors are to be altered. The two fields update the color selections for the selected layer. **Borders/highlight** shows the color used to trace around the edge of each polygon on the layer. This is also the color used in illuminated sections of the 3D mask displays. **Fill/shadow** is the color used to fill each polygon on the layer. It is also the color used to indicate shadowed areas of masks on the 3D mask displays.

10.8.9: Notes On Monochrome Operation

All MASKVIEWS features are available in monochrome mode. This is selected automatically for systems not supporting enough colors or by using the `-mono` command line switch. Differences that are seen are that all key values displayed and listed are in terms of pattern fills instead of colored regions. When you select a mask for editing in **Phases** or **Transmittances** mode, all other masks will appear as outlines only, instead of dimmer color

A.1: Introduction

Models and Algorithms used by one dimensional (1D) electrical solvers in DECKBUILD and TONYPLOT.

Note: This appendix is intended to serve as a quick reference only. A detailed description of the semiconductor device physical models is provided in the ATLAS manual.

1D electrical solvers, available by using the `extract` command in DECKBUILD or in TONYPLOT, are based on the iterative solution of the Poisson equation:

$$\text{div}(\varepsilon \nabla \psi) = q(p - n + N_D^+ - N_A^-) - \rho_F \quad \text{A-1}$$

where ψ is the potential, ε is the dielectrical permittivity, n and p are the electron and hole concentrations, and ρ_F is the fixed charge.

QUICKBIP uses the continuity equations to calculate n and p :

$$\frac{1}{q} \text{div} J_n^{\otimes} - U_n^{\otimes} = 0 \quad \text{A-2}$$

$$\frac{1}{q} \text{div} J_p^{\otimes} - U_p^{\otimes} = 0 \quad \text{A-3}$$

where:

$$J_n^{\otimes} = q \mu_n E_n^{\otimes} \cdot n + q D_n \nabla n \quad \text{A-4}$$

$$J_p^{\otimes} = q \mu_p E_p^{\otimes} \cdot p + q D_p \nabla p \quad \text{A-5}$$

$$D_n = \frac{kT}{q} \mu_n, \quad D_p = \frac{kT}{q} \mu_p \quad \text{A-6}$$

A.1.1: Physical Models

All electrical solvers take into account the following models and effects:

- Temperature dependence, such as kT/q or E_g
- Concentration-dependent mobility (with built-in temperature dependence)
- Field-dependent mobility (perpendicular field with built-in temperature dependence)
- Material work function (for MOS structures)
- Fixed interface charge

A.2: Concentration Dependent Mobility

The concentration dependent mobilities for n and p respectively are:

$$\mu_n^D = \mu_{nmin} + \frac{\Delta\mu_n}{1 + N_{total}/N_{nref}} \quad \text{A-7}$$

$$\mu_p^D = \mu_{pmin} + \frac{\Delta\mu_p}{1 + N_{total}/N_{pref}} \quad \text{A-8}$$

where:

$$\mu_{nmin} = 88 \cdot \left(\frac{Y}{300}\right)^{-0.57} \quad \text{A-9}$$

$$\mu_{pmin} = 54.3 \cdot \left(\frac{Y}{300}\right)^{-0.57} \quad \text{A-10}$$

$$\Delta\mu_n = 1252 \cdot \left(\frac{Y}{300}\right)^{-2.33} \quad \text{A-11}$$

$$\Delta\mu_p = 407 \cdot \left(\frac{Y}{300}\right)^{-2.33} \quad \text{A-12}$$

$$N_{nref} = 1.432 \cdot 10^{17} \left(\frac{Y}{300}\right)^{2.456} \quad \text{A-13}$$

$$N_{pref} = 2.67 \cdot 10^{17} \left(\frac{Y}{300}\right)^{2.456} \quad \text{A-14}$$

A.3: Field Dependent Mobility Model

The field dependent mobilities for n and p respectively are:

$$\mu_n = \frac{\mu_n^D}{\sqrt{1 + 1.54 \cdot 10^{-5} \cdot E}} \quad \text{A-15}$$

$$\mu_p = \frac{\mu_p^D}{\sqrt{1 + 5.35 \cdot 10^{-5} \cdot E}} \quad \text{A-16}$$

A.4: Sheet Resistance Calculation

After solving the Poisson equation, the sheet resistance for each semiconductor layer is estimated using:

$$R_{sh} = \int_{xleft}^{xright} \frac{dx}{q\mu_n \cdot n + q\mu_p \cdot p} \quad \text{A-17}$$

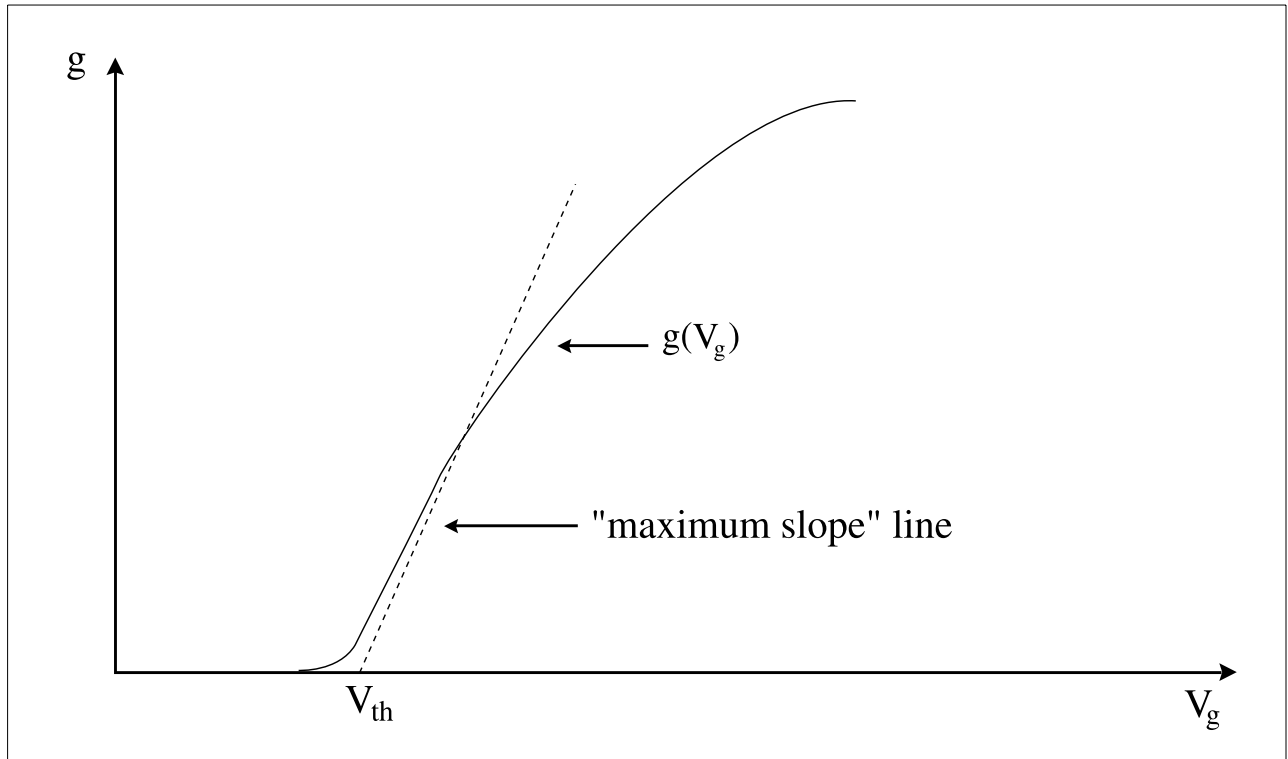
$xleft$ and $xright$ are determined by the p - n junction locations and the semiconductor material boundaries.

A.5: Threshold Voltage Calculation

Threshold voltage calculation is based on the calculated sheet resistance. In MOS mode (1-D vt extraction), the solver will calculate threshold voltage automatically. First, the conductance of the channel region will be calculated for each gate voltage applied. If an NMOSFET structure is assumed, then:

$$g(V_g) = \int_O^{x_{inv}} (q\mu_n)/n dx \quad A-18$$

O corresponds to the oxide-silicon interface and x_{inv} is the boundary of the inversion layer. Threshold voltage will be determined using the $g(V_g)$ curve as an intersection with the V_g axis of the straight line drawn through two points on the $g(V_g)$ curve, corresponding to the maximum slope region shown below.



A.5.1: Breakdown Voltage Calculation

Breakdown voltage calculation is based on estimation of ionization integrals for electrons and holes. Breakdown is determined by the condition that one of the integrals is greater than 1. The ionization rates are calculated using the following equations (See the Selberherr model in the ATLAS manual):

$$\alpha_n = AN \cdot \exp\left[-\left(\frac{BN}{E}\right)^{BETAN}\right] \quad A-19$$

$$\alpha_p = AP \cdot \exp\left[-\left(\frac{BP}{E}\right)^{BETAP}\right] \quad A-20$$

where:

```
AN = AN1 if E < EGRANAN = AN2 if E > EGRAN
AP = AP1 if E < EGRANAP = AP2 if E > EGRAN
BN = BN1 if E < EGRANBN = BN2 if E > EGRAN
BP = BP1 if E < EGRANBP = BP2 if E > EGRAN
```

The values of the parameters AN1, AN2, AP1, AP2, BN1, BN2, BP1, BP2, BETAN, BETAP, EGRAN are user-definable (through the `extract` command or pop-up menu). Their default values are:

```
AN1=7.03e5 cm-1
AN2=7.03e5 cm-1
BN1=1.231e6 V/cm
BN2=1.231e6 V/cm
AP1=6.71e5 cm-1
AP2=1.582e6 cm-1
BP1=1.693e6 V/cm
BP2=2.036e6 V/cm
BETAN=1.0 (unitless)
BETAP=1.0 (unitless)
EGRAN=4e5 V/cm
```

B.1: DBInternal

DBINTERNAL is a simple but powerful DECKBUILD tool that allows you to create a Design Of Experiments (DOE) from a pair of input files. Amongst other things, you can create corner models for process parameters or device characteristics or both.

Any parameters that are to be used as variables must be specified as `set` statements in a template file. Any results of interest should be calculated using `extract` statements.

The DOE is specified with simple `sweep` statements in a separate design file. The `sweep` statement defines which variables are required in the DOE, and the range of values these variables are to take.

The parameter values and the results of each simulation can be stored in a file that can be viewed in TONYPLOT or used as a data base for input to a statistical analysis tool such as SPAYN.

B.1.1: Example

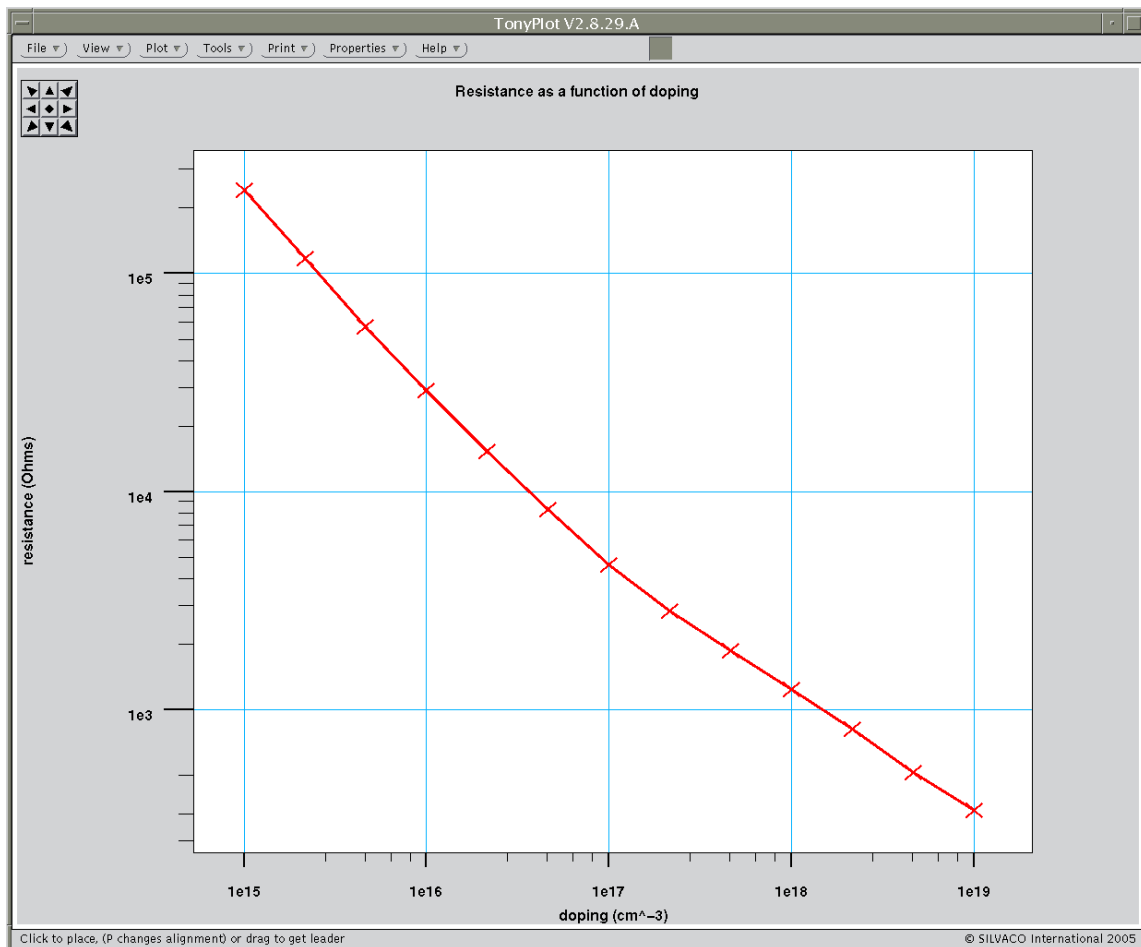
Suppose you have an ATLAS deck (`resistor_template.in`) for a simple resistor (the doping is controlled by a `set` statement). When run, this deck calculates the resistance from the gradient of the VI curve.

```
go atlas
set doping=1e16
mesh width=2
x.mesh loc=0.0 spac=0.25
x.mesh loc=1.0 spac=0.25
y.mesh loc=0.0 spac=0.25
y.mesh loc=10.0 spac=0.25
region num=1 silicon
electrode num=1 top name=ground
electrode num=2 bottom name=anode
doping uniform n.type conc=$doping
models conmob
solve init
log outf=dop$'doping'.log
solve vanode=0.0 vstep=0.1 vfinal=2.0 name=anode
log off
extract init infile="dop$'doping'.log"
extract name="res" grad from curve(i."anode",v."anode") where y.val=1
quit
```

If you want to investigate how doping affects the resistance, you can create a DBINTERNAL deck (sweep.in) that defines an experiment (a series of trials). In this example, the doping is changing between 10^{15} and 10^{19} cm^{-3} (at three points per decade, thirteen points in all).

```
go internal
load infile=resistor_template.in
sweep parameter=doping type=power range="1.0e15, 1.0e19, 13"
save type=sdb outfile=resistance.dat
quit
```

When you execute sweep.in, DBINTERNAL runs the resistor_template.in deck several times, each time changing the value on the set doping= line. DBINTERNAL also collates the data generated by the extract name="res" line and saves it in sdb format suitable for viewing with TONYPLOT. The resistance as a function of doping is shown in the following figure.



B.2: The Template File

The template file is a description of the class of simulations you want to perform. It should be a deck that will execute correctly when run within DECKBUILD.

Any variables that you need DBINTERNAL to control must be defined on a `set` line. For example, the file `resistor_template.in` has the line

```
set doping=1e16
```

so DBINTERNAL can change the value of the variable `doping`. DBINTERNAL ignores the actual value on the `set` line in the template file. It is safe to set variables that DBINTERNAL doesn't control. They will remain with the value defined in the template file.

The variable is normally used to set numbers in the template file. For example:

```
doping uniform n.type conc=$doping
```

where the doping concentration is being set by the variable `doping`. But it is also useful to be able to use the value as a string in a filename. In this instance, you should enclose the variable name in single quotes. For example:

```
log outf=dop$doping'.log
extract init infile="dop$doping'.log"
```

So if `doping` had been set to `1e16`, the filename would be `"dop1e16.log"`.

The template file may have `extract` statements. For example:

```
extract init infile="dop$doping'.log"
extract name="res" grad from curve(i."anode",v."anode") where y.val=1
```

DBINTERNAL will recognize you are interested in the result `"res"` (for example) and will collect these results from each simulation.

B.3: The Experiment File

The experiment file has three main parts

B.3.1: Load command

```
load infile=resistor_template.in
```

This tells DBINTERNAL which file to use as the basis for the simulations.

B.3.2: Experiment command

```
sweep parameter=doping type=power range="1.0e15, 1.0e19, 13"
```

This tells DBINTERNAL how you want the variables to change.

B.3.3: Save Command

If the template file contains extract statements, you also want a save command

```
save type=sdb outfile=resistance.dat
```

which tells DBINTERNAL where to save the extracted data. The saved file will contain the values of all the independent variables (the variables defined in the experiment command) and the values of all the dependent variables (the variables calculated with extract statements).

B.4: Technical Details

DBINTERNAL reads in the template file and looks for any variables defined on an `extract` line and makes a note of their names. The name must be the first parameter after the `extract` command and have no spaces.

```
extract name="res" ...
```

DBINTERNAL also knows what parameters have been set on the `experiment` line. For example, DBInternal will control the parameters `x` and `y`.

```
sweep parameter=x type=linear range=1,2,2 \
      parameter=y type=linear range=3,5,3
```

To run a trial, DBINTERNAL creates a temporary `<infile>` with the name

```
<infile>=dbinternal_temporary_<name>_<pid>
```

`<name>` is the name of the machine and `<pid>` is the program ID of the DBINTERNAL program. (See also the “`log`” command). The temporary file is a copy of the template file with different values on any `set` line that correspond to a parameter in the `experiment` command. For instance, if the template file had the lines

```
set x=5
set y=10
set z=15
```

and you ran the earlier `sweep` command the first temporary file would have the lines

```
set x=1
set y=3
set z=15
```

DBINTERNAL will change the values for parameters it recognizes and leaves the other `set` lines alone.

DBINTERNAL then runs a trial by producing a child (DECKBUILD) with the command

```
deckbuild [-int][-ascii][-noplot] -run <infile> -outfile <infile>.out
```

Once DECKBUILD is finished, DBINTERNAL will parse the `<infile>.out` file to find the values generated by the `extract` statements. The `<infile>` and the `<infile>.out` are then deleted.

This procedure (creating an `<infile>`, starting DECKBUILD, examining the `<infile>.out`) is repeated for the remaining sets of parameters in the experiment.

B.5: DBInternal Commands

DBINTERNAL commands are generally of the form

```
<command> <param1>=<value1> <param2>=<value2> ...
```

The commands and the parameters may be abbreviated but they must be long enough to be recognized. For instance, the save command may be shortened to sa, but not to s because that would not distinguish between save and sweep.

If the value contains whitespace, it must be enclosed in quotes. For instance

```
range="1.0e15, 1.0e19, 13"
```

But if the value is a single block of text, there is no need for the quotes. This range can be entered as

```
range=1.0e15,1.0e19,13
```

DBINTERNAL recognizes the following commands.

B.5.1: doe

Syntax

```
doe type=<doe_type> \  
    parameter=<param1> range="center, delta" \  
    parameter=<param2> range="center, delta"
```

Description

The trials of a DOE experiment correspond to various points on or near a hypercube around some origin in parameter space. The results can be used to create a model for the dependent variables over the hypercube.

For example, suppose you had a process that generated FETs with gate lengths in the range 95-105 μm and recess depths in the range 45-55 μm . The parameter space is a square with corners (95, 45), (95, 55), (105, 45) and (105, 55). Suppose you want to know how breakdown voltage is affected by these parameters. If you knew there were a simple linear relationship, you could simulate at the midpoint and the two points where the axis intersected with the hypercube ((100, 50), (105, 50), (100, 55)) and fit a simple linear model through the results. If you thought there were a more complex relationship, you would simulate at more points over the hypercube. For example, the midpoint and all the corners.

A parameter should be the name of a variable in the template deck. The names (e.g., <param1>) cannot be abbreviated. They must be exactly as they appear in the template deck.

The range is two numbers. The first is center of the hypercube (i.e., the value of the parameter at the middle of its range). The second is the distance from the center to the edge of the hypercube (or half the range of the parameter).

The points are almost always high symmetry points on the hypercube, such as the corners of the hypercube, the points where an axis intersects the hypercube, and a midpoint along an edge of the hypercube. The best way to see the points generated by a DOE type is to use the no_exec command and setting the range of the parameters to "0, 1". Therefore, 0 indicates a center point, 1 indicates one side of the hypercube and -1 the other side.

Example

```
no_exec outfile=tlff.dat
doe type=two_level_full_factorial \
    parameter=p1 range=0,1 \
    parameter=p2 range=0,1 \
    parameter=p3 range=0,1
```

DOE Types

The DOE type must one of the following:

- `gradient_analysis`
- `two_level_full_factorial`
- `two_level_half_factorial`
- `three_level_full_factorial`
- `face_centered_cubic`
- `circumscribed_circle`
- `box_behnken`

gradient_analysis

This type does a simulation at the center point and at the points one positive step along each axis. An experiment with N parameters has $N+1$ trials.

two_level_full_factorial

This type does a simulation corresponding to every node of the N -dimensional hypercube. An experiment with N parameters has 2^N trials.

two_level_half_factorial

This type does half the simulations of the `two_level_full_factorial` and no two nodes are on the same edge. An experiment with N parameters has 2^{N-1} trials.

three_level_full_factorial

This type does a simulation corresponding to every node and every half point of the N -dimensional hypercube. An experiment with N parameters has 3^N trials.

face_centered_cubic

This type does a simulation corresponding to every node. Every point where an axis intersects the hypercube and the center point. An experiment with N parameters has $2^N + 2N + 1$ trials.

circumscribed_circle

This type is similar to the `face_centered_cubic` type but the axis points now lie on the surface of the hypersphere that passes through the hypercube node points (i.e., all points are equidistant from the origin). An experiment with N parameters has $2^N + 2N + 1$ trials.

box_behnken

The design matrix for this simulation is based on a balanced or partially balanced block design. There is no easy relationship between the number of parameters and the number of trials in the experiment. For example, an experiment with seven parameters requires 57 trials and an experiment with nine parameters requires 97 trials. But an experiment with eight parameters is particularly inefficient and requires 225 trials.

B.5.2: endsave

Syntax

```
endsave
```

Description

This command tells DBInternal to stop saving data to the current file. See the `save` command for more information.

Example

```
save type=sdb outfile=example.out  
sweep parameter=doping type=power range=1e15,1e19,13  
endsave
```

This stops DBINTERNAL from trying to save any more data to `example.out`.

B.5.3: log

Syntax

```
log outfile=<filename_root>
```

Description

This command tells DBINTERNAL to keep the output generated by the child DECKBUILD for each trial. When generating the temporary input file for a trial, DBINTERNAL uses the `<filename_root>` and appends the ID of the trial (for example, the first trial has an ID of zero and the second trial has an ID of one). The usual command is issued to run the trial

```
deckbuild -int -noplot -run <infile> -outfile <infile>.out
```

At the end of the trial, only the `<infile>` is deleted and the `<infile>.out` will remain.

Example

```
load infile=example.in  
log outfile=keep  
sweep parameter=doping type=linear range=1,4,4
```

This will generate the files `keep1.out`, `keep2.out`, `keep3.out`, and `keep4.out`.

B.5.4: monte_carlo

Syntax

```
monte_carlo number=<num_trials> \  
    parameter=<param1> type=<mc_type> coeffs="list, of, numbers" \  
    parameter=<param2> type=<mc_type> coeffs="list, of, numbers"
```

Description

The `monte_carlo` command generates an experiment from a specified number of trials. All parameter values in each trial are random. Each parameter is drawn from its own distribution.

The number is the number of trials you want in the experiment.

A parameter should be the name of a variable in the template deck. The names (e.g., `<param1>`) cannot be abbreviated. They must be exactly as they appear in the template deck.

The type must be one of the following:

- uniform
- normal
- log_normal
- gamma
- weibull

These are the types of random distributions a parameter will be extracted from. The coeffs are a list of numbers that describe the random distribution, the number and meaning of the coeffs depending upon which distribution was chosen.

Random Distributions

The probability density function and the required coefficients for the following random distributions.

uniform

The uniform type takes random numbers evenly spaced between two limits. The probability density function is

$$p(x) = 0 \quad (x < x_{lo} \text{ or } x \geq x_{hi})$$

$$p(x) = \frac{1}{x_{hi} - x_{lo}} \quad (x_{lo} \leq x < x_{hi}) \quad \text{B-1}$$

This distribution needs two coefficients, the minimum and maximum allowed values

$$coeffs = "x_{lo}, x_{hi}" \quad \text{B-2}$$

normal

The normal type is a Gaussian probability density. The probability density function is

$$p(x) = \frac{1}{\sqrt{2\pi} \cdot \sigma} \exp\left(-\frac{(x - \mu)^2}{2\sigma^2}\right) \quad \text{B-3}$$

This distribution needs two coefficients, the mean and the standard deviation.

$$coeffs = "\mu, \sigma" \quad \text{B-4}$$

log_normal

The log_normal type has a probability density function of (the distribution of log(x) would be normal)

$$p(x) = \frac{1}{\sqrt{2\pi} \cdot X\sigma} \exp\left(-\frac{(\log(X) - \mu)^2}{2\sigma^2}\right) \quad \text{with} \quad X = \pm(x - \tau) \quad \text{B-5}$$

This distribution needs four coefficients (the last coefficient should be +1 or -1 and gives the sign in front of X).

$$coeffs = "a \mu, \sigma, \pm" \quad \text{B-6}$$

gamma

The gamma type is the time to wait for several events that occur with a Poisson distribution. The probability density function is

$$p(x) = X^{a-1} \cdot \left(-\frac{\exp(X)}{\sigma \cdot \Gamma(a)} \right) \quad \text{with} \quad X = \pm \left(\frac{x - \mu}{\sigma} \right). \quad \text{B-7}$$

This distribution needs four coefficients (the first coefficient should be a +ve integer and the last should be +1 or -1).

coeffs = "a μ, σ, ± " B-8

Weibull

The probability density function of the weibull type is

$$p(x) = \frac{a}{\sigma} \cdot X^{a-1} \cdot \exp(-X^a) \quad \text{with} \quad X = \pm \left(\frac{x - \mu}{\sigma} \right) \quad \text{B-9}$$

This distribution needs four coefficients (the last coefficient should be +1 or -1).

coeffs = "a μ, σ, ± "

B.5.5: no_exec**Syntax**

```
no_exec type=<ssf|spayn> outfile=<filename>
```

Description

This is a debugging command. If it is issued before an experiment, then DBINTERNAL will generate a file <filename>. This file contains a list of the independent variables, which would have been passed to the simulations. No actual trials will be performed. An ssf format file is a simple list of numbers suitable for viewing in a text editor. It can also be plotted in TONYPLOT. A spayn format file can be analyzed with SPAYN.

Example

If you run an experiment file

```
no_exec type=ssf outfile=example.out
sweep parameter=length type=linear range=1,2,2 \
parameter=width type=linear range=1,3,3
```

the data in example.out would be

```
...
0 1 1
1 2 1
2 1 2
3 2 2
4 1 3
5 2 3
```

The first column is the ordinal of the trial. The second column is the length value of that trial (e.g., 1 or 2). The third column is the width value of that trial (e.g., 1, 2 or 3).

B.5.6: option

Syntax

```
option [[!]int][[!]ascii] [[!]plot]
```

Description

The `option` command controls some of the command line options passed to DECKBUILD when running a trial.

The value of a parameter (`int`, `ascii`, or `plot`) is either true or false. If the parameter is present, it is set to true. If it is negated (with the `!`), it is set to false. If it is absent, it is set to a default value. The default for `int` is true. The default for both `ascii` and `plot` is false.

A trial is run with the command

```
deckbuild [-int] [-ascii] [-noplot] -run <infile> -outfile <infile>.out
```

The `-int` option tells DECKBUILD to start DBINTERNAL as the default simulator. The `-ascii` option tells DECKBUILD not to start its GUI. The `-noplot` option tells DECKBUILD to ignore any TONYPLOT commands in `<infile>`.

The `-int` option will be passed if `int` is true and will be absent if `int` is false.

The `-ascii` option will be passed if `ascii` is true and will be absent if `ascii` is false.

The `-noplot` option will be passed if `plot` is false and will be absent if `plot` is true.

Example

```
option ascii !plot
```

This will run the trials without the DECKBUILD GUI and ignoring any TONYPLOT commands in the trial deck.

```
option !ascii plot
```

This will run the trials inside a child DECKBUILD GUI and will execute any TONYPLOT commands in the trial deck.

B.5.7: save

Syntax

```
save type=<sdb|spayn> outfile=<filename>
```

Description

The `save` command saves the data generated by the experiment in the file `<filename>`. You can output the data in `sdb` format (to be viewed in TONYPLOT) or in `spayn` format (to be analyzed by SPAYN).

The following data is stored for each trial:

- The ID of the trial.
- The values of the parameters defined on the experiment line.
- The values of the parameters calculated with `extract` commands.

You can place the `save` command before or after the experiment line. If it comes before the experiment line, the file will be rewritten at the end of each trial. Therefore if something unforeseen hap-

pens during an experiment, you will have the data from the trials that were completed. If it comes after the experiment line, all the data from the experiment will be written at once.

Only one file at a time can be active. If you define two `save` statements before an experiment line, only the second will actually get the following data.

```
save type=sdb outfile=save.sdb

save type=spayn outfile=save.spayn

sweep parameter=doping type=power range=1e15,1e19,13
```

The file `save.sdb` will have no data (the file `save.spayn` will be fine). Once a file is active, it will remain active until a subsequent `save` command makes a different file active or until an `endsave` command is given. A file, however, will only have data from one experiment.

Warning: If you have more than one experiment line in a deck, be very careful with the `save` command or you will lose data. For example, the following is the wrong way.

```
sweep parameter=doping type=power range=1e15,1e19,13

save type=sdb outfile=one.sdb

sweep parameter=doping type=linear range=1e16,1e17,11

save type=sdb outfile=two.sdb
```

This will perform the first experiment, then save that experiment to `one.sdb`, then perform the second experiment. But because `one.sdb` is still the active file, DBINTERNAL will write the data from the second experiment to `one.sdb`, destroying the data that was already there. At the end of this run, both `one.sdb` and `two.sdb` will contain the same data. In this case, you must use the `endsave` command to tell DBINTERNAL to deactivate the active file.

Example

```
sweep parameter=doping type=power range=1e15,1e19,13

save type=sdb outfile=one.sdb

endsave

sweep parameter=doping type=linear range=1e16,1e17,11

save type=sdb outfile=two.sdb
```

B.5.8: sweep

Syntax

```
sweep parameter=<param1> type=<sweep_type> range="start, stop, num" \
      parameter=<param2> type=<sweep_type> data="point ... list"
```

Description

The `sweep` command generates an experiment from all combinations of individual parameter values. The first parameter changes with the highest frequency. The final parameter changes with the lowest frequency.

A parameter should be the name of a variable in the template deck. The names (e.g., `<param1>`) cannot be abbreviated. They must be exactly as they appear in the template deck.

The type must be either `linear`, `power` or `list`.

The range is three numbers, the initial value of the parameter, the final value of the parameter, and the number of points. This is used to for the linear and power types.

In a linear sweep, the parameter values are evenly spaced.

Example 1

```
sweep parameter=x type=linear range="1,4,7"
```

This generates for x the values:

- 1
- 1.5
- 2
- 2.5
- 3
- 3.5
- 4

In a power sweep, the log of the parameter values are evenly spaced.

Example 2

```
sweep parameter=y type=power range="1e10, 1e15, 6"
```

This generates for y the values:

- 1e10
- 1e11
- 1e12
- 1e13
- 1e14
- 1e15

The data is a list of values to assign to the parameter.

Example 3

```
sweep parameter=z type=list data="1,2.5,3,3.1,4"
```

This assigns each of the values (one at a time) to z. The number of trials in the experiment is the product of the number of points for each parameter.

Example 4

```
sweep parameter=x type=linear range="1,4,7" \  
parameter=y type=power range="1e10, 1e15, 6"
```

This generates an experiment with 42 trials. All the values of x in combination with all the values of y. Adding another variable:

```
sweep parameter=x type=linear range="1,4,7" \  
parameter=y type=power range="1e10, 1e15, 6" \  
parameter=z type=list data="0, 2, 3"
```

would increase the number of trials to 126. All 42 trials from the previous experiment with z=0, and all 42 trials with z=2, and all 42 trials with z=3.

This page is intentionally left blank.

Numerics

1D RSM Graphs	
Show data	7-29
2D Mesh Plot Display	
3D	7-24
Contours	7-20
Junction	7-24
Light	7-23
Lines	7-25
Regions	7-20
Vectors	7-22
<i>See also</i> Plot Display	
2D RSM Contours	
Contour Type	7-30
Mesh	7-30
Output Range	7-30
Projection	7-30
X and Y Quantities	7-30
Z Quantity	7-30
3D Plot Control	
Rotating	7-18
Scaling	7-18
<i>See also</i> Plot Control	
3D RSM Surfaces	7-30

A

Adding Targets	
Curve Values	6-14
Point Values	6-14
<i>See also</i> Targets	
Advanced Features (Automation Tools)	
Experimental Design	1-7
Integrated Graphics	1-8
Intelligent Execution	1-7
Networked Execution	1-8
On-Line Help	1-8
Process Flow Editor	1-7
Quality Printouts	1-8
Response Surface Analysis	1-8
Sensitivity Analysis	1-7
Split Points, Experimental Trees, and Worksheets	1-7
VWF Database	1-7
Worksheets	6-24
Advanced Features (Meshing Example 2)	
3D Structures	9-29
Circular Devices	9-32
Combining Two ATHENA Structures into a Single Device	9-30
Stretch and Cut	9-31
Advanced Topics (DeckBuild)	
Extraction	4-3
Generic Decks	4-3
Analytic Functions	
Location Dependent Variables	9-51

User Supplied Variables. <i>See also</i> Roll-Off Functions	9-51
Annotation	
Axis Labels	7-34
Footers	7-34
Macro	7-34
Range	7-34
Showing Plots	7-34
Special Characters	7-34
Statistics Plots	7-34
Titles	7-33
<i>See also</i> TonyPlot	
ATHENA	2-24, 2-25, 4-31 10-1
Grid Definition	10-15
Layout Experiments	10-16
<i>See also</i> Device Simulation, Device Structure , and	
Process Simulators	
ATLAS	2-24, 2-25 8-26, 9-6
<i>See also</i> Device Simulation and Device Structure	
Auto Interfacing	4-40
<i>See also</i> DeckBuild	

B

Base Window	
Control Panel	9-4
Control Windows	9-4
File Menu	7-7
Help Menu	7-11
Layout and Functionality	9-3
Main Panel Controls	9-4
Plot menu	7-9
Print Menu	7-10
Production Menu	7-10
Properties Menu	7-11
Tools menu	7-9
View Menu	7-9
<i>See also</i> DevEdit and TonyPlot	
Bipolar Extract	
QUICKBIP	5-40
BJT	5-32
<i>See also</i> Device Extraction	
Box Plots	7-32
<i>See also</i> Statistics Display	
Breakdown Voltage Calculation	A-5

C

Calculation	
Breakdown Voltage	A-5
Sheet Resistance	A-4
Threshold Voltage	A-5
Camera	
Depth Cue	8-34
Projection	8-34

Scaling	8-34
Cartesian Graphs	
Group	7-27
Scales	7-26
X Quantity	7-26
Y Quantities	7-27
Channel Surface Impurity Concentration	2-23
Clever	4-32
Colors	7-62, 7-63
General	7-61
Sequence	7-63
Structure	7-62
<i>See also</i> Properties	
Command File	
Default Files	9-6
<i>See also</i> File Control	
Command Stream	
Finishing TPCS	7-72
Help	7-72
Concentration Dependent Mobility	A-2
Conductance vs. Bias	2-20
Contours	
Data Constraint Settings	8-12
<i>See also</i> Display Modes	
Controlling Plots. <i>See</i> 3D Plot Control and Plot Control	
Cross Section Display	7-28
Curved Target	
Creating	6-14
Creating from a Data File	6-15
Curves	
Abs Operator with Axis	5-34
Ave Operator	5-33
Axis Manipulation Combined with Max and Abs Operators	5-35
Axis Manipulation Combined with Y Value Intercept	5-35
Axis Manipulation with Constants	5-34
Creation	5-33
Data Format File Extract with X Limits	5-35
Derivative	5-35
Gradient at Axis Intercept	5-34
Impurity Transform against Depth	5-35
Max Operator	5-33
Max Operator with Axis Intercept	5-34
Min Operator	5-33
Min Operator with Axis Intercept	5-34
Second Intercept Occurrence	5-34
X Axis Interception of Line Created by Maxslope Operator	5-34
X Value Intercept for Specified Y	5-33
Y Axis Interception of Line Created by Minslope Operator	5-35
Y Value Intercept for Specified X	5-33
Customized Extract Statements	
DEFAULTS	5-20
Syntax	5-7
<i>See also</i> Extract	
Cutline	
Control Items	7-38
Creating	7-39
Creating Multiple Plots	7-39
Cross-Section	7-39

Deleting	7-39
Loading	4-35
Movies	7-40
Shifting	7-40

D

Data Visualization. *See* TonyPlot, TonyPlot 3, and TonyPlot 3D

Deck Writing Paradigm	4-30
DeckBuild	1-3, 2-7, 4-1, 10-2
Auto Interfacing	4-40
Commands	4-30
Controls	4-14
Environment Variables	4-69
Error Messages	4-70
Extract	5-1
<i>See also</i> Extract	
Features	4-1
History	4-38
IC Layout Interface	4-43
Internal Interface	4-57
Main Control	4-20
Optimizer	6-1
<i>See also</i> Optimizer	
Purpose	4-1
Quitting	4-10
Remote Simulation	4-58
Running A Deck	4-8
Smartspice Interface	4-56
Starting	4-5, 4-11
Tools	4-33
UTMOST Interface	4-48
Writing SSUPREM3 Input Deck	4-5
DeckBuild Commands	
ASSIGN	4-60-63
AUTOELECTRODE	4-63-64
Clever	4-32
DEFINE	4-64-65
ELSE	4-67
EXTRACT	4-65
GO	4-65-66
IF	4-67-68
IF.END	4-67
L.END	4-67-68
L.MODIFY	4-67-68
LOOP	4-67-68
MASK	4-68
MASKVIEWS	4-69-70
Mercury	4-32
Parsing	4-30
Process Simulators	4-31-32
SET	4-70-72
SOURCE	4-72-73
STMT	4-73-74
SYSTEM	4-74-75
TONYPLOT	4-75
UNDEFINE	4-64
DeckBuild Commands	
<i>See also</i> DeckBuild	
DeckBuild Controls	

Main Window	4-14	Device Extraction	5-29
Text Subwindow	4-15	<i>See also</i> Extract	
TTY Subwindow	4-18	Device Physics	
DeckBuild Features		Defining	2-25
Advanced Topics	4-3	Device Simulation	2-25
Autointerface	4-2	Defining Physics	2-25
Examples and Tutorial	4-3	Extracting Parameters	2-27
Execution Control	4-3	Running	2-25
Optimization	4-3	Device Structure	
Simulators	4-2	Completing	2-24
DeckBuild Main Window		Device Threshold Voltage	2-20
Execution Control Buttons	4-15	Display Modes	
Menu Buttons	4-14	Contours	8-12
DevEdit	1-4	Isosurface	8-16
Base Window	9-3	Rays	8-14
<i>See also</i> Base Window		Regions	8-11
DOPING	9-39	Vectors	8-18
Editing Regions	9-35	Display Modes	8-9
FILE CONTROL	9-5	<i>See also</i> TonyPlot3D	
IMPURITIES	9-46	Displaying Plots. <i>See</i> Plot Display	
Meshing	9-7	Doping	
Problems	9-1	3D DOPING	9-40
REGIONS	9-36	DELETING SOURCE OBJECTS	9-40
ROLL-OFF FUNCTION	9-51	IMPURITY SOURCE BOX	9-39
Solution	9-1	IMPURITY SOURCE LINE	9-39
Startup	9-2	SOURCE ATTRIBUTES	9-40
STATEMENTS	9-59	Doping Profiles	
STRUCTURE EDITING	9-34	Adding	9-53
When Not to Use	9-1	Draw Modes	8-9
When to use	9-1	Drawing Regions	
DevEdit Statements		Base Impurity (Doping)	9-37
BASE.MESH	9-60		
BOUNDARY.CONDITIONING	9-60	E	
CONSTRAINT.MESH	9-62	Editing Regions	
CUT	9-65	ADDING	9-35
DEPOSIT	9-65	SELECTING MATERIALS	9-35
FLIP	9-66	Electrodes	4-44
IMPURITY	9-67	esponse	1-8
IMPURITY REFINER	9-71	Execution	1-7, 4-28
INITIALIZE	9-72	Execution Control	
JOIN	9-73	Breakpoints	4-28
MESH	9-73	Buttons	4-28
MIRROR	9-74	Concepts	4-27
MOVE	9-75	<i>See also</i> DeckBuild	
PROFILE	9-76	Initializing the Simulator	4-29
QUIT	9-77	Pausing, Stopping, and Restarting the Simulator	4-29
REFINE	9-77	Setting Lines	4-28
REGION	9-78	Execution Control Buttons	
RENUMBER.REGIONS	9-80	Stepping Through and Running the Deck	4-28
SOURCE	9-80	Experimental Design	1-7
STRETCH	9-80	<i>See also</i> Advanced Features (Automation Tools)	
STRUCTURE	9-83	Experimental Results	
SUBSTRATE	9-84	Control Items	7-91
WORK.AREA	9-84	Uses	7-92
Z.PLANES	9-85	Extract	5-1, 5-38
Device Extraction		Customized Statements	5-7
BJT	5-32		
Curve	5-29		
Curve Manipulation	5-31		

G

1

J

Join Function	
User Supplied Variables	9-54

K

Key Legends	
Drawing Styles	7-71
Positioning Key Boxes	7-70
Types	7-70
<i>See also</i> TonyPlot and TonyPlot 3	

L

Labels	
Control Items	7-35
Placing	7-36
Special	7-37
<i>See also</i> TonyPlot	
Loading	2-5
Loading Examples	2-5

M

Macros	
Function	7-77
Main DeckBuild Controls	
Arguments	4-26
Choosing a Simulator	4-20
Control Pad	4-20
Execution Control	4-27
Formatting	4-26
Messages	4-25
Options	4-23
Simulator Controls	4-21
Simulator Properties	4-21
Start Simulator	4-21
Main Window	
Menu Options	8-4
Plot Control Toolbar	8-7
Plot Control Using the Mouse	8-7
Toolbar	8-6
<i>See also</i> TonyPlot3D	
Manager	1-5
Applications	3-4
Changing Directories	3-2
Customizing Attributes	3-3
File and Application Windows	3-2
File types	3-5
On Line Help	3-2
Starting	3-1
Starting Applications	3-2
Mask	
Misalignment and CD Experimentation	4-63
Mask Editing	
Drawing Objects	10-9
Layout Layers	10-8
Object Editing	10-12
Parameters	10-7

Polygon	10-11
User Defined Objects	10-10
Mask Statements	4-43
Mask View Properties	
3D Mask	10-27
MaskView Utilities	
Order Layers	10-24
MaskViews	1-4, 4-63
ATHENA	10-1, 10-13
Editing	10-7
<i>See also</i> Mask Editing	
Example Layout File	10-3
Files	10-18
Interfacing With A Simulator	10-3
Main Window	10-5
OPTOLITH	10-17
Optolith	10-1
Properties	10-26
Simulator	10-13
SSUPREM3	10-1, 10-13
Starting	4-34
Starting from DeckBuild	10-2
Starting from UNIX Prompt	10-4
Starting Inside The VWF	10-4
Utilities	10-22
MaskViews Files	
Cutline Files	10-19
GDS2 & CIF Import/Export	10-19
GDSII Stream Format	10-21
Loading and Saving	10-18
OPTOLITH Image File	10-21
MaskViews Main Window	
Control Panel	10-6
Key Panel	10-6
Layout and Functionality	10-5
MaskViews Main Window	10-5
<i>See also</i> MaskViews	
MaskViews Properties	
Color	10-28
Default	10-26
Display	10-27
Drag and Drop	10-27
Import	10-28
Monochrome Operation	10-28
Printer	10-28
MaskViews Utilities	
On-Line Help	10-25
Printouts	10-25
Regions	10-22
Release Notes	10-25
Rescaling	10-23
Zoom and Pan	10-23
Mercury	4-32
Mesh. <i>See</i> Mesh Creation	
Mesh Creation	
Base Mesh Parameters	9-45
Final Meshing	9-45
Impurity Refinement	9-45

Mesh Constraints	9-45
MESH CONTROLS	9-45
Saving	9-45
Mesh Creation (Meshing Example 1)	
Manual Refine Box	9-17
Mesh Constraints	9-16
Mesh Parameters	9-15
MeshBuild	9-15
Refine on Quantities	9-15
Mesh Creation (Meshing Example 2)	
Boundary Conditioning	9-22
Mesh Constraints	9-25
Mesh Parameters	9-24
Refine on Quantities	9-24
MeshBuild	
3D STRUCTURES	9-44
ADAPTIVE MESHING	9-43
BOUNDARY CONDITIONING	9-41
Limitations	9-42
MESH CONSTRAINTS	9-42
REFINEMENT	9-43
Refining	9-44
Relaxing	9-44
TENSOR	9-44
WORK AREA RESIZING	9-44
Meshing	9-7
Creating	9-8
Mesh Creation	9-45
MeshBuild	9-41
Meshing Example 1	9-8
Meshing Example 2	9-20
REMESHING	9-20
Meshing Display (Meshing Example 2)	
Doping	9-21
Impurity Junctions	9-21
Zoom	9-21
Meshing Example 1	
Defining Regions	9-9
Impurities	9-13
Mesh Creation	9-15
Saving Mesh Files	9-17
Work Area	9-8
Meshing Example 2	
Advanced Features	9-29
Display	9-21
Loading the Structure	9-20
Mesh Creation	9-22
Obtain existing structure	9-20
Saving Mesh Files	9-27
Structure editing	9-20
Summary	9-33
Miscellaneous	7-67
MOS Device Tests	5-36
Multiple Plots	7-39

N

Networked Execution	1-8
<i>See also</i> Advanced Features (Automation Tools)	

O

Object Editor	
Hierarchy Structure	8-21
Mouse Action Functionality	8-22
Right Mouse Menus	8-23
Objects	
Editing	10-12
<i>See also</i> Mask Editing	
Polygons	10-9
Serifs	10-10
Sticks	10-10
User-Defined	10-10
On-Line Help	1-8
<i>See also</i> Advanced Features (Automation Tools)	
Optimizer	1-4, 4-3, 6-1
Features	6-1
File I/O	6-26
Graphics	6-21
Optimization Modes	6-3
Optimization Tuning	6-30
Optimizer Window	6-2
Parameters	6-4
Printing	6-28
Results	6-23
Setup	6-20
Targets	6-12
Terminology	6-1
Worksheet Editing	6-24
Optimizer Parameter	
Defaults	6-10
Editing	6-7
Optimizer Parameter Editing	
Numeric Values	6-7
Parameter Name	6-8
Response Type	6-7
Optimizer Parameters	
Adding	6-4
Copying	6-10
Deleting	6-6
Enabling/Disabling	6-10
Folding Columns	6-11
Linked	6-9
Optimizer Results	
Sensitivity	6-23
Optimizer Setup	
Editing	6-20
Saving	6-20
OPTOLITH	
Image File	10-21
Phase and Transmittance Values	10-17
Optolith	10-1
Outline	7-86
Overlays	
Controlling	7-84
Creating	7-84
Cutlines	7-85
Displaying	7-84

Identifying Data	7-84	Form Editor	7-51
Level Names	7-85	Options	7-49
Properties	7-85	Printer Queues	7-52
Splitting	7-84	Queues	7-52
See also TonyPlot		Setting Print Options	7-54
		System Configuration	7-52
		See also Printing Editor and TonyPlot	
P		Process Extraction	
Parameter Type		Curves	5-5
BOOLEAN	9-86	Entering Statements	5-4
COLOR	9-86	Examples	5-21
IMPURITY	9-87	Process Extraction Examples	
MATERIAL	9-94	1D Material Region Boundary	5-24
PATTERN	9-99	1D Max/Min Concentration	5-23
Phase and Transmittance Values		2D Concentration Area	5-24
Phase Display Mode	10-17	2D Concentration File	5-24
Serifs	10-17	2D Material Region Boundary	5-24
Transmittance Display Mode	10-17	2D Max/Min Concentration	5-23
Physical Models	A-1	2D Maximum Concentration File	5-24
Pie Charts	7-32	ED Tree (Optolith)	5-28
See also Statistics Display		Elapsed time	5-28
Plot	2-9, 7-32	Electrical Concentration Curve	5-28
Plot Control		Junction Breakdown Curve	5-26
Mouse	8-7	Junction Capacitance Curve	5-25
Pointer Zooming	7-13	Junction Depth	5-21
Selecting Plots	7-13	Material Thickness	5-21
Toolbar	8-7	QUICKMOS 1D Vt	5-22
Plot Display		QUICKMOS CV Curve	5-25
2D Mesh	7-19	Sheet Conductance	5-22
See also 2D Mesh Plot Display		Sheet Resistance	5-22
Plots	2-15, 2-17	Sheet Resistance/Conductance Bias Curves	5-27
Plotting		SIMS Curve	5-26
2-D Structures	2-16	SRP Curve	5-27
Contour	2-17	Surface Concentration	5-21
Cutline	2-18	Process Flow Editor. See also Advanced Features	1-7
Modifying. See also TonyPlot	2-9	Process Simulation	2-7
Overlaying	2-15	Comparing Structure Files	2-13
Printing	2-11	Continuing	2-15
Plotting The Current Structure	4-9	Contour Plots	2-17
Pointer	7-45	Cutline Plots	2-18
Poisson Solver	7-47	Extracting Parameters	2-19
Applying A Bias To A Layer	7-48	History Function	2-12
Setup	7-48	Overlaying Plots	2-15
Polar Charts	7-27	Plotting 2-D Structures	2-16
Positioning Key Boxes	7-70	Plotting Structures	2-8
Printer		Running	2-7
Form Editor	7-51	Using TonyPlot	2-9
Printer Drivers		Process Simulators	
Stack Size. See also TonyPlot	7-55	Selecting Categories	4-32
Printing Editor		Writing Process Input Decks	4-31
Controls	7-51	Writing Text	4-32
Printers	7-50	Production Mode	
Printing Plots		ASA Setup	7-93
Adding Forms to TonyPlots	7-53	Enabling	7-86
Adding Printers to TonyPlot	7-52	Experimental Results	7-91
At Startup	7-52	Failure Analysis	7-87
Editor	7-50	Input Distributions	7-90
		Input Parameter Ranges	7-89
		Input Sliders	7-87

Interactive RSM Control	7-87
Outline	7-86
Popup	7-86
SPC Limits	7-91
Synthesis	7-88
Yield Analysis	7-89
Production Modes	
Optimizer Setup	7-92

Q

Quality Printouts	1-8
<i>See also</i> Advanced Features (Automation Tools)	
QUICKBIP	
Bipolar Extract	5-40
<i>See also</i> Extract	

R

REGIONS	
Deleting	9-37
Deleting Boundary Points	9-38
Drawing	9-36
Modifying	9-38
Regions	
Editing	9-35
Electrodes	4-44
Regions (Meshing Example 1)	
Adding	9-9
Electrodes	9-13
Etch then Adding	9-12
Material Selection and Uniform Doping	9-10
Modifying	9-12
Setting Mole (Composition) Fraction	9-11
Remote Simulation	
Options	4-58
Troubleshooting	4-58
Remote Simulation	4-58
<i>See also</i> DeckBuild	
Resetting The Current Line	4-8
Response Surface Analysis	1-8
<i>See also</i> Advanced Features (Automation Tools)	
Result	7-88
Roll-Off Directions	
Rolloff=Both	9-48
Rolloff=High	9-48
Rolloff=High.P.Step	9-49
Rolloff=Low	9-49
Rolloff=Low.P.Step	9-50
Rolloff=P.Step	9-50
Rolloff=Step	9-49
Rolloff=Step.P.High	9-50
Rolloff=Step.P.Low	9-49
Roll-Off Functions	
ANALYTIC FUNCTIONS	9-51
COMBINING IMPURITY ROLLOFFS	9-55
DELETING IMPURITIES	9-55
DOPING PROFILES	9-53
EDITING IMPURITIES	9-55

RSM Display	
1D Graphs	7-29
<i>See also</i> 1D RSM Graphs	
1D mode	7-29
2D Contours	7-30
<i>See also</i> 2D RSM Contours	
3D Surfaces	7-30
<i>See also</i> 3D RSM Surfaces	
Rules	4-46

S

Saving Mesh Files (Meshing Example 1)	
Batch Mode	9-18
Command File	9-18
Structure File	9-17
Saving Mesh Files (Meshing Example 2)	
Batch Mode	9-27
Command File	9-27
Structure File	9-27
Saving The Deck	4-9
Scatter Plots	
X Axis	7-32
Y Axis	7-32
<i>See also</i> Statistics Display	
Semiconductor Process Data Base	1-5
Sensitivity Analysis	1-7
<i>See also</i> Advanced Features (Automation Tools)	
Sequence	
Colors	7-63
Set Files	
Creating	7-82
Loading	7-82
Syntax	7-83
<i>See also</i> TonyPlot	
Sheet Resistance	
Beneath Oxide Spacer	2-23
LLD	2-23
n++ Source/Drain	2-22
Sheet Resistance Calculation	A-4
Sheet Resistances	2-22
Show	7-29
Silvaco Standard Structure File (.str)	
Loading	9-5
Saving	9-5
Simulation	
Running	2-7
Stepping Through	2-7
<i>See also</i> Process Simulation	
SmartSpice	
Interface	4-56
Smith Charts	7-27
Source Attributes	
Gaussian	9-40
Source/Drain	2-19
SPC Limits	
Control Items	7-91

Uses	7-91	Saving Changes	4-17
SPDB	1-5	See also DeckBuild Controls	
Split Points, Experimental Trees and Worksheets	1-7	Threshold Voltage Calculation	A-5
See also Advanced Features (Automation Tools)		TonyPlot	1-4, 2-9, 7-1, 8-2
Splitting	7-84	2-D Structures	2-16
SSUPREM3	4-5, 10-1	3D Plot Control	7-18
See also DeckBuild		Adding Forms	7-53
Statistics Display		Adding Printers	7-52
Box Plots	7-32	Annotation	7-33
Histograms	7-31	See also Annotation	
Pie Charts	7-32	Base Window	7-7
See also Pie Charts		See also Base Window	
Scatter Plots	7-32	Command Stream	7-15, 7-72
Sunray Plots	7-32	Examining Data	7-1
Stop At Function	2-15	File Loader. See File Loader	
Stream	10-21	Fonts	7-66
Structure Editing		Functions	7-76
EDITING SUMMARY	9-34	Key Commands	7-14
PANNING	9-34	Key Legends	7-70
Resolution	9-34	Labels. See also Labels	7-35
ZOOMING	9-34	Modifying Plots	2-9
Structures	2-16	Online Help	7-2
Subwindow	4-69	Overlaying	2-15
Sunray Plots	7-32	Overlays	7-84
See also Statistics Display		Panning	2-11
Surface Analysis		Plot Control	7-13
Response	1-8	See also Plot Control	
See also RSM Display		Plot Display. See Plot Display	
Surfaces	7-30	Plot Menu	7-16
Synthesis		Print Options	7-49
Results	7-88	Printer Drivers	7-55
Setup	7-88	Printing Plots	2-11, 7-49
		Production Mode	7-86
		Properties	7-56
		RSM Display	7-29
		See also RSM Display	
		Set Files	7-82
		Standard Controls	7-2
		Starting	4-33
		Starting	7-4
		Statistics Display	7-31
		Structure Files	2-13
		Terminology	7-2
		Tools. See also Tools	7-38
		User Data Files	7-80
		X-Y Graphic Display	7-26
		Zooming	2-11
		TonyPlot Properties	
		Drawing Options	7-56
		Environment	7-65
		Fonts	7-66
		Functions	7-69
		General Colors	7-61
		Key Options	7-64
		Materials	7-68
		Miscellaneous	7-67
		Overlays	7-60
		Plot Options	7-57
		Sequence Colors	7-63
		Sequence Marks	7-64

<i>See also TonyPlot</i>	
TonyPlot3D	8-2
Display Modes	8-9
Main Window	8-3
Operating Platforms	8-40
Plot Control	8-7
Properties	8-34
Starting	8-2
Tools	8-20
TonyPlot3D Properties	
Camera	8-34
Color	8-35
Fonts	8-35
Legend	8-36
Lights	8-37
Main Window	7-58
Mouse	8-38
Structure	8-39
Structure Colors	7-62
Tool Settings	7-59
<i>See also TonyPlot3D</i>	
Tools	
Outline	7-38
Cutplane	8-26
HP 4145 Emulator	7-43
Integration	7-44
Manager	4-37
Maskviews	4-34
Movie	7-42
Object Editor	8-20
Poisson Solver	7-47
Probe	7-41, 8-30
Ruler	7-40, 8-32
Text Editor	4-37
TonyPlot. <i>See also TonyPlot</i>	4-33
Tracers	7-46
<i>See also DeckBuild, Manager, MaskViews, TonyPlot, and TonyPlot3D</i>	
TPCS	
Finishing	7-72
Function	7-79
Syntax	7-72
Tracers	7-46
Setup	7-47
TTY Subwindow	
Cut, Paste, and Copy	4-18
Editing	4-18
Entering Commands	4-18
Error Messages	4-19, 4-69
Saving/Resetting	4-19
U	
User Data File Format Examples	
Display	7-81
Equation	7-81
Transistor	7-81
User Data Files	
Creating	7-80
Format	7-80

Loading	7-80
<i>See also TonyPlot</i>	
User Data Format	
Examples	7-81
<i>See also TonyPlot</i>	
UTMOST	1-4
Input Deck	4-48
Interface	4-48

V

Virtual Wafer Fab (VWF)	
Introduction	1-1
VWF Automation Tools	
Advanced Features	1-7
Experimentation	1-6
MASTER Database	1-7
Productivity Enhancements	1-8
Virtual Wafer FAB	1-6
VWF Database	1-7
<i>See also Advanced Features (Automation Tools)</i>	
VWF Interactive Tools	
DeckBuild	1-3
DevEdit	1-4
Getting Started	1-3
MaskViews	1-4
TonyPlot	1-4

W

Worksheet	
Mouseless Operation	6-25
Numeric Values	6-24
Selecting Rows	6-24
<i>See also Optimizer</i>	

X

X-Y Graphic Display	
Cartesian Graphs	7-26
<i>See also Cartesian Graphs</i>	
Cross Section Display	7-28
<i>See also Cross Section Display</i>	
Polar Charts	7-27
<i>See also Polar Charts</i>	
Smith Charts	7-27
<i>See also Smith Charts</i>	

Y

Yield Analysis	
Description	7-89
Results	7-89
Setup	7-89